Títol: Servei de promoció d'establiments mitjançant geolocalització

Alumne: Juan Soler Company
Director/Ponent: Andrés Manso / Maria Ribera Sancho

Departament: ESSI

Data: 05/06/2012

DADES DEL PROJECTE

Títol del Projecte: Servei de promoció d'establiments mitjançant geolocalització

Nom de l'estudiant: Juan Soler Company

Titulació: Enginyeria Informàtica

Crèdits: 37.5

Director/Ponent: Andrés Manso / Maria Ribera Sancho

Departament: ESSI

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President:

Vocal:

Secretari:

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

ÍNDEX

Agraïments

En el desenvolupament d'aquest projecte, hi ha moltes persones que han col·laborat per tal que arribés a bon port. En alguns casos, l'ajuda ha estat de caire tècnic, en altres, més de caire filosòfic.

Abans de res, he d'agrair als meus pares, que sempre hagin confiat en mi, en els bons i en els mals moments, gràcies al seu suport he arribat aquí. Només pel fet de fer que es sentin orgullosos, tota aquesta experiència haurà valgut la pena. A les meves germanes, cunyats i nebots, també els he d'esmentar, perquè formen part de tota aquesta gent imprescindible dins la meva vida, que tot i que pensin que no els tinc en compte, sempre són al meu cap i al meu cor.

No parlar dels meus amics seria un exercici d'estupidesa vertaderament gran. Al sr. Massi se li ha d'agrair que hagi aguantat tants d'anys essent el meu millor amic, i pràcticament essent part de la meva família. En Mateu m'ha ensenyat, que sa perseverança i sa paciència, són dues de ses claus de s'èxit en aquesta vida, i en Raúl, que mai no ens hem de rendir, encara que sembli que és s'única opció.
Pau, Borja, Gabino, Vicent, Pedro, Enric, Juen, Yudes... tots els abans esmentats i ells, han d'aparèixer, ja que, tot i no tenir-ne ni idea d'informàtica, formen part d'aquest gremi de gent especial, originària de Menorca, que fa que aquella roca perduda enmig de la mar, sigui el lloc on més a gust estic del món.

No esmentar a les meves amigues, a part d'injust, seria perillós . He d'agrair a na Esther, que hagi passat tant de temps amb jo i m'hagi aguantat en els moments frikis en els quals només podia parlar de programació. Sandra, Rubia, Paula … totes mereixeu el meu reconeixement, ja que mantenir amistat amb mi, no ha de ser una tasca trivial.

En un caire més tècnic, he d'esmentar els meus amics de la facultat, destacant el Guille, ambaixador andorrà, que tants moments ha passat amb mi, aguantant els meus acudits dolents, i el meu mal humor matinal. Mr.X, Grido, Roland, Cristo, Marc, tots vosaltres mereixeu aparèixer, ja que la vostra ajuda, m'ha fet arribar aquí.

De la feina, el Guillem ha de ser destacat, ja que m'ha ensenyat pràcticament tot el que sé sobre programació web i absolutament tot, sobre administració de servidors, a part de ser una gran persona i una font il·limitada de coneixements tècnics. Al "caido" Lluís, li agraeixo els bons moments que vam passar, i als altres programadors, com al Met, la Mónica, Juan, Joel, el bon ambient que generen.

Agraeixo sobretot als creadors de Stackoverflow, que hagin creat el lloc on hi ha resolts tots els dubtes possibles. També dono gràcies al mal funcionament dels servidors de battle.net, ja que han ajudat que el Diablo 3 no em boicotegés el desenvolupament del projecte.

Introducció

Vivim en un món on la tecnologia és imprescindible. Internet, la xarxa de xarxes, ha passat de ser un recurs innovador, a ser una necessitat per a la gran majoria de la població.

Internet ha aportat una capacitat de comunicació espectacular. Les notícies, fa uns anys, t'arribaven o bé pel telenotícies, per la ràdio o pel diari. La difusió de les notícies era tan ràpida com els mitjans abans esmentats. Avui dia, tota aquesta informació es transmet de manera quasi instantània a nivell mundial. Si combinem aquesta difusió amb la quantitat de dispositius amb capacitat de consumir la informació, veiem com la societat viu molt informada.

Smartphones, tablets, portàtils, eBooks... a dia d'avui la gran majoria de dispositius té accés a Internet ja sigui via Wifi, 3G o per la via que sigui. El fet és que els Terabytes i Terabytes d'informació que circulen per Internet, poden esser consultats per la gran majoria. A més, mitjançant les xarxes socials, comunicar-se mai no havia estat tan fàcil.

El següent pas d'aquesta evolució constant, ja no és que la informació estigui disponible per si la vols consultar, sinó que la informació t'arribi directament, i amb els smartphones ha estat possible. Gràcies al fet que compten amb mecanismes per a descobrir la teva localització geogràfica, han sorgit gran quantitat d'aplicacions que exploten aquesta informació, ja sigui per a mostrar-te com arribar a algun lloc o per donar-te informació sobre el teu voltant. Es podria dir que la generació dels smartphones és incapaç de perdre's.

Hi ha hagut empreses que s'han aprofitat d'aquestes circumstàncies per a oferir serveis de publicitat geolocalitzada, és a dir, si hom entra en certes zones, li arriba publicitat d'algun establiment del seu voltant. Això s'ha anomenat GeoFencing, i empreses com Placecast, ho han començat a promocionar als Estats Units.

A nivell personal, considero la Geolocalització un concepte apassionant. Entrar en un establiment, del tipus que sigui, sense tenir clar que la nostra necessitat restarà satisfeta en sortir per la porta és un risc que de cada vegada s'està tornant més poc desitjable.
A l'actualitat, la gent vol explotar al màxim el seu temps, i si vol comprar-se l'últim gadget que ha sortit al mercat i entra en un establiment en què no el tenen en stock, aquest fet provocarà el seu enuig a causa del temps perdut. L'obtenció de la ubicació de l'usuari pot fer que el que abans trigaríem una tarda a solucionar, ho solucionem en mitja hora.

A més, l'actual recessió econòmica, combinada amb la gran evolució d'Internet ha fet que neixi el que s'anomena el "Smart Shopper", el consumidor del segle XXI, que no vol renunciar a productes de qualitat, però que s'ajuda de la xarxa per a trobar els millors preus possibles.

Amb el meu projecte, s'intentarà utilitzar la geolocalització perquè els usuaris puguin tenir informació sobre el seu voltant. En aquest cas, les dades obtingudes seran sobre restaurants que hi ha al voltant, i sobre ofertes. D'aquesta manera, un usuari que no sap on sopar, pot conèixer quines opcions té al seu voltant.

Context del Projecte

Donat que el projecte s'ha desenvolupat en una empresa, considero important descriure breument el tipus d'empresa i una mica els tipus de projectes que es desenvolupen.

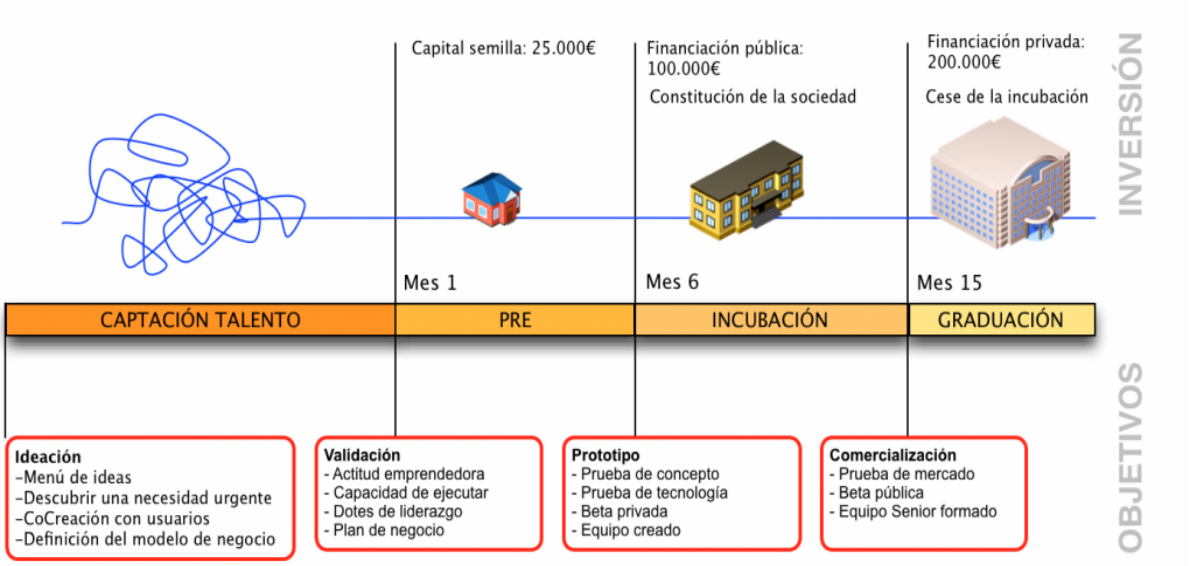
L'empresa

El projecte s'ha desenvolupat en una empresa anomenada **Appstylus S.L** , localitzada dins l'edifici "Almogàvers Business Factory", edifici que pertany a Barcelona Activa, l'agència de desenvolupament local de l'Ajuntament de Barcelona. El que es promou des de Barcelona Activa és la creació d'empreses i l'emprenedoria, la innovació i en general, la creació d'ocupació.

Situats en aquest marc, es podria dir que l'empresa en què treballo es tracta d'una incubadora d'empreses, situada dins una incubadora d'empreses (Barcelona Activa).

Appstylus és una incubadora d'empreses en la qual s'orienta i es dóna suport ,en tots els processos necessaris, als emprenedors perquè una idea es converteixi en un nou servei en línia que cobreixi una necessitat trobada en un segment concret del mercat.

El procés d'incubació de projectes que es segueix en l'empresa es pot representar amb el següent gràfic:



Com es pot observar, es segueix tota una sèrie de fases fins al desenvolupament d'un projecte. Inicialment es fa una captació d'idees (es fan "Brainstormings" periòdicament), que posteriorment es discuteixen, per tal de veure si realment tenen sentit. Se'n descarten moltes, però les que es consideren bones, es queden en el que anomenem un procés de preincubació.

En aquest punt, un dels emprenedors de l'empresa comença a desenvolupar la idea, i crea un primer pla de negoci, unes especificacions més precises, i defineix a quin segment del mercat s'orientarà el projecte.

Posteriorment, es passa a la fase d'incubació, en què es desenvoluparà una primera versió de l'aplicació, en forma de prototip, que s'anirà avaluant periòdicament per usuaris, els quals aportaran un "feedback" que serà crucial, per a adaptar al màxim el producte als seus interessos. Es diu que un projecte passa a la fase de Graduació, quan la incubació ha finalitzat, fent proves amb usuaris i coneixent les seves opinions. En aquell punt es desenvoluparà l'aplicació definitiva i es comercialitzarà.

Com hem esmentat abans, l'empresa és una incubadora. D'aquesta manera, quan s'està en la fase de la incubació, el projecte passa a constituir-se com una societat, la qual tot i dependre encara d'Appstylus en un inici, pot arribar a transformar-se en una entitat pràcticament independent si s'obtenen els beneficis necessaris.

L'empresa està formada en un 80% per estudiants, considerats emprenedors, que estan assignats a un o més projectes dels que s'estan desenvolupant. En tot moment els estudiants tenen llibertat per a proposar noves idees per a futurs projectes de l'empresa.

A dia d'avui a l'empresa hi ha quinze treballadors en total, i hi ha quatre projectes en desenvolupament, alguns en fases més avançades que altres.

Aquests quatre projectes tenen a veure amb tecnologies web o tecnologies mòbils. Concretament, els projectes que hi ha en desenvolupament a dia d'avui són aquests:

Groupiest

Groupiest permet a l'usuari triar una sèrie de temes que siguin del seu interès, i el que es generarà serà un site amb informació extreta de diverses fonts de la xarxa, relacionades amb els temes d'interès de l'usuari. D'aquesta manera, si l'usuari és un apassionat dels videojocs, tindrà a una web gran quantitat d'informació de diferents tipus (vídeos, imatges, notícies, links...), sobre videojocs.

NotedLinks

NotedLinks és una eina que s'encarrega d'enriquir el contingut d'una web. El que fa és agafar el text de la teva web, i analitzar-lo, de tal manera que si troba informació relacionada amb els termes del text, se't mostra.

Zeed Protection

Aquest projecte té un focus diferent, ofereix un servei que detecta vulnerabilitats a webs. Permetrà contractar el servei per part d'empreses externes que vulguin veure si el seu web és segur, i en el cas de no ser-ho, seran informats de quins són els problemes de seguretat que tenen.

Chollos Cercanos (nom no definitiu)

En aquest projecte estarà emmarcat el meu projecte de final de carrera. Aquest projecte el que vol aconseguir és utilitzar les tecnologies mòbils per a apropar els responsables dels establiments als seus potencials clients utilitzant la geolocalització. Una persona que vol promocionar el seu establiment es podrà enregistrar en un portal web, on serà capaç de publicar actualitzacions, ofertes i promocions en el seu perfil. L'usuari haurà d'indicar la posició geogràfica del seu establiment, per tal que les seves publicacions arribin als usuaris d'smartphones que es trobin a prop del seu establiment.

Per altra banda, hi haurà una aplicació mòbil que mostrarà a l'usuari de l'smartphone les promocions, ofertes i actualitzacions d'establiments que té al seu entorn. L'objectiu és que l'usuari pugui configurar l'aplicació de tal manera que els avisos que rebi s'adaptin al màxim als seus interessos.

El públic objectiu de l'aplicació mòbil són aquells usuaris d'smartphones que, en moltes ocasions, saben el que volen comprar, però que no tenen preferències definides pel que fa al lloc on fer-ho. Aquesta gent, si s'instal·la l'aplicació, obtindrà informació dels establiments del voltant, la qual cosa pot ser que faciliti la seva decisió.

Per altra banda, tot aquell establiment que vulgui promocionar-se, serà públic objectiu del portal web que s'ha esmentat.

Poso l'exemple de les ofertes, perquè pot ser el més bàsic i fàcil d'entendre, però com veiem, es podria aplicar en altres contextos.

Per exemple, potser un grup de joves sap on vol sortir de festa, però no sap on. Segurament la rutina prengui la decisió per ells, descartant aquell lloc nou que acaben de obrir, per pur desconeixement. Potser aquests joves reben una notificació amb les cançons que estan sonant en aquest nou local, i canvien d'opinió de cop.

Això seria una descripció breu del que es pretén aconseguir finalment. Evidentment, el projecte està en un estat molt inicial de desenvolupament, i moltes coses poden canviar encara respecte a allò que acabarà essent el producte final. El projecte encara està en la fase de preincubació.

La part que es desenvoluparà i es presentarà com a projecte de final de carrera serà primer un estudi dels usuaris utilitzant les xarxes socials per a fer-ho, d'una manera merament observacional en un inici i posteriorment d'una manera més agressiva.

Finalment es desenvoluparà el que es pot considerar un primer prototip d'aplicació mòbil, que utilitzarà una sèrie d'ofertes geolocalitzades, i mostrarà a l'usuari quines són les que té més a prop. Servirà primer per a formar-me en les tecnologies que utilitzem, i per a fer un primer prototip que treballi amb ofertes externes, que pugui ser fàcilment ampliat amb les que recollim mitjançant el portal web del que he parlat abans, i que es desenvoluparà a posteriori (i que queda fora de l'abast del meu projecte).

Tota aquesta part la desenvoluparé sense més ajuda que la que m'aportarà el director del meu projecte (que s'encarregarà de supervisar la seva evolució).

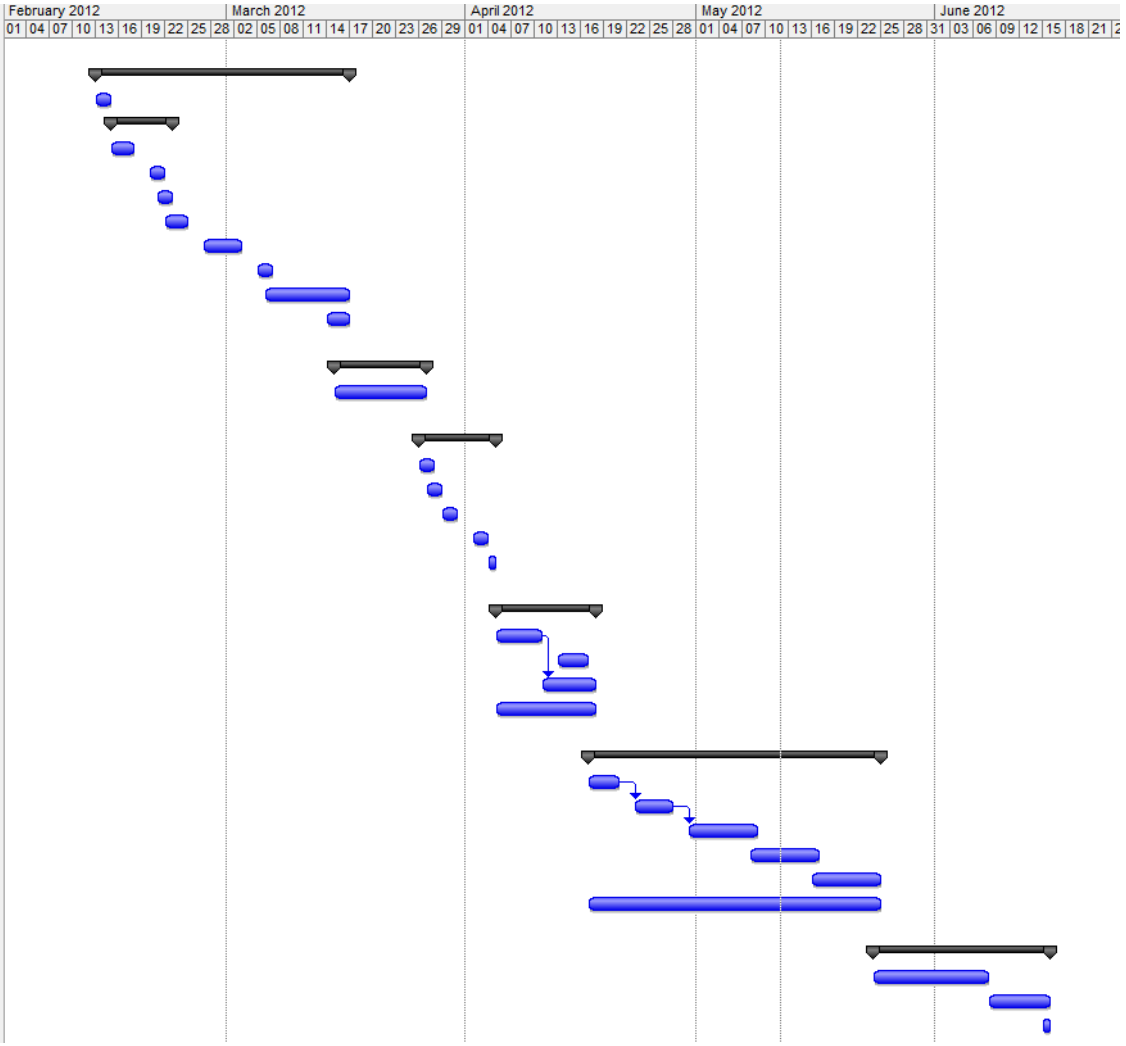
Cal destacar que treballaré tant en el projecte descrit, com en el projecte Groupiest (que hem descrit breument abans), projecte amb el qual porto vinculat un temps.

El procés amb què s'ha arribat a desenvolupar aquest projecte, va sorgir d'unes idees que vaig proposar al director de l'empresa, que tenien a veure amb geolocalització, i amb el fet que els usuaris poguessin conèixer informació sobre el que tenen al voltant. Una vegada exposades les idees, es van discutir i van anar evolucionant.
Finalment es van celebrar una sèrie de Brainstormings en els quals es va acabar polint la idea, fins a arribar a allò que he descrit.

Es podria dir que el projecte està en preincubació, i com hem explicat abans, es sol fer el prototip més endavant. En aquest cas, s'avançarà el prototipatge per al desenvolupament del meu PFC, i s'aprofitarà per a formar-me, de tal manera que una vegada la idea estigui del tot definida, ja tindré tots els coneixements necessaris per a desenvolupar el projecte final de l'empresa, de manera eficient.

Planificació

	Task Name	Duration	Start	Finish
1				
2	**Estudis Previs**	**25 days?**	**Mon 13/02/12**	**Fri 16/03/12**
3	Configuració de l'entorn	2 days?	Mon 13/02/12	Tue 14/02/12
4	**Desenvolupament Crawler**	**6 days?**	**Wed 15/02/12**	**Wed 22/02/12**
5	Groupon	3 days?	Wed 15/02/12	Fri 17/02/12
6	Groupalia	2 days?	Mon 20/02/12	Tue 21/02/12
7	El Tenedor	2 days?	Tue 21/02/12	Wed 22/02/12
8	Comunicació amb Restalo	3 days?	Wed 22/02/12	Fri 24/02/12
9	Recopilació Automatitzada de Tweets	5 days?	Mon 27/02/12	Fri 02/03/12
10	Representació Tweets API Google Maps	2 days?	Mon 05/03/12	Tue 06/03/12
11	Twitter Bot	9 days?	Tue 06/03/12	Fri 16/03/12
12	Redacció Informe Previ	3 days?	Wed 14/03/12	Fri 16/03/12
13				
14	**Anàlisi dels resultats dels estudis**	**8 days**	**Thu 15/03/12**	**Mon 26/03/12**
15	Documentar Estudis previs	8 days	Thu 15/03/12	Mon 26/03/12
16				
17	**Anàlisi de tecnologies mòbils**	**8 days?**	**Mon 26/03/12**	**Wed 04/04/12**
18	Prova de Rendiment de Jquery Mobile	2 days?	Mon 26/03/12	Tue 27/03/12
19	Prova de Rendiment de Sencha Touch	2 days?	Tue 27/03/12	Wed 28/03/12
20	Recerca de informació sobre l'SDK d'Android	2 days?	Thu 29/03/12	Fri 30/03/12
21	Recerca d'informació sobre l'SDK d'iOS	2 days?	Mon 02/04/12	Tue 03/04/12
22	Documentar l'anàlisi de tecnologies	1 day?	Wed 04/04/12	Wed 04/04/12
23				
24	**Especificació I Disseny de l'aplicació**	**9 days?**	**Thu 05/04/12**	**Tue 17/04/12**
25	Definició concreta dels casos d'us.	4 days	Thu 05/04/12	Tue 10/04/12
26	Disseny de l'interficie	2 days?	Fri 13/04/12	Mon 16/04/12
27	Disseny de les connexions amb el servidor	5 days?	Wed 11/04/12	Tue 17/04/12
28	Documentació Especificació I Disseny	9 days	Thu 05/04/12	Tue 17/04/12
29				
30	**Implementació de l'aplicació**	**28 days?**	**Tue 17/04/12**	**Thu 24/05/12**
31	Estudi de la tecnologia escollida	4 days?	Tue 17/04/12	Fri 20/04/12
32	Implementació de la part del servidor	5 days	Mon 23/04/12	Fri 27/04/12
33	Implementació de l'interfície	7 days	Mon 30/04/12	Tue 08/05/12
34	Implementació de les connexions amb el servido	7 days	Tue 08/05/12	Wed 16/05/12
35	Depuració I Testeig	7 days	Wed 16/05/12	Thu 24/05/12
36	Documentació Implementació	28 days	Tue 17/04/12	Thu 24/05/12
37				
38	**Tancament**	**17 days?**	**Thu 24/05/12**	**Fri 15/06/12**
39	Acabar la memòria	11 days?	Thu 24/05/12	Thu 07/06/12
40	Preparar Presenació	6 days?	Fri 08/06/12	Fri 15/06/12
41	Presentació	1 day?	Fri 15/06/12	Fri 15/06/12



S'acaba de presentar un diagrama de Gantt amb la planificació del projecte. Podem observar que s'ha dividit en diverses fases:

- Uns **Estudis Previs** on s'ha intentat analitzar el comportament dels usuaris, utilitzant la xarxa social Twitter, com a eina per a fer-ho. També s'han recopilat tant ofertes com informació sobre restaurants.

- Una **Anàlisi dels Estudis Previs** on s'agafaran els resultats, tant de la recopilació de Tweets, com els del Bot, i s'analitzaran i documentaran, per a extreure'n algunes conclusions.

- Una **Anàlisi de Tecnologies Mòbils** , l'objectiu del qual serà molt clar, veure quina de les tecnologies que s'han escollit com a candidates convé més per al desenvolupament del projecte.

- L'**Especificació i Disseny del projecte** , fase on s'especificarà i dissenyarà el sistema de manera detallada, per a facilitar el posterior procés d'implementació.

- La **Implementació** del mateix, on realment es materialitzarà tot el que s'ha especificat anteriorment.

- I per últim una fase de **Tancament**, fase on s'acabarà de redactar la memòria, i es prepararà la defensa del projecte.

Cal destacar que hi ha tasques que al diagrama surten en paral·lel, com són algunes de les tasques de documentació. Això s'expressa així per una raó molt simple, com és que no es vol documentar les fases de cop (salvant el cas dels estudis previs), sinó que l'objectiu és anar documentant tal i com es van desenvolupant les tasques per a obtenir una documentació precisa.

Pressupost

A continuació es podrà veure un pressupost d'aquest projecte. S'ha de destacar que tot i que el treball d'analista, arquitecte i programador l'he fet jo mateix, serà considerat com tres apartats diferents, pel fet que no són tasques comparables, ni remunerables de la mateixa manera. A part d'aquests tres rols, també s'afegeixen les hores on l'administrador de sistemes de l'empresa m'ha ajudat en la configuració inicial del servidor.

A més, també es compten les hores del director del projecte i del director tècnic de l'empresa, persones que han col·laborat per tal que aquest projecte arribés a bon port. Com veurem a continuació, la seva col·laboració s'ha basat en les reunions setmanals que s'han fet per al control de l'evolució del projecte. Amb el director tècnic, aquestes reunions han estat setmanals, amb el director del projecte, s'han fet dues vegades a la setmana.

Com a comentari, cal destacar que totes les hores en les quals s'han desenvolupat els estudis previs i les hores de la fase de tancament que s'ha vist en el Gantt, s'han assignat a l'analista.

Recurs	Preu/hora	Hores	Descripció	Import
Administrador de sistemes	30	4	Configuració inicial del servidor.	120
Director Tècnic	40	17	Reunions Setmanals	680
Director del Projecte	60	34	Reunions dues vegades per setmana	2040
Analista	25	488	Fase d'especificació, Disseny, estudis previs i Tancament	12200
Arquitecte	30	64	Fase d'elecció de les tecnologies	1920
Programador	15	224	Fase d'implementació	3360
				20320

Es pot observar que el cost estimat del projecte, comptant les hores especificades al Gantt (tenint en compte una jornada laboral de 8 hores), i afegint la figura de l'administrador de sistemes, el director tècnic i el director de projecte, és de **20320 €**

Estudis Previs

En aquest projecte, en una fase prèvia, s'ha plantejat la necessitat de conèixer les necessitats i preferències dels usuaris, per tal de dissenyar a posteriori una aplicació adaptada als gustos dels seus usuaris potencials.

Per a desenvolupar aquesta tasca, s'han seguit una sèrie de passes. Inicialment, es va voler acotar el segment de mercat al qual atacar, per tal d'estar en un mercat conegut, amb unes característiques propícies perquè l'aplicació fos útil, i sobretot, utilitzada.

Es va decidir que l'ideal seria intentar arribar a la gent de Catalunya, ja que es podria dir que els coneixem una mica, i es pot intuir com reaccionarien de manera més o menys precisa.

Aquesta fase es podria segmentar en 3 parts diferenciades:

- Recopilació d'ofertes geolocalitzades

- Estudi de les zones on més s'utilitzen els smartphones

- Promoció d'ofertes per Twitter

A continuació, s'explicarà en detall aquestes tres fases.

Recopilació d'ofertes geolocalitzades

El que es va fer inicialment va ser buscar diferents fonts d'ofertes , centrant-nos principalment en la ciutat de Barcelona per qüestions purament demogràfiques. És més fàcil, trobar ofertes a la ciutat de Barcelona que a Sant Cugat, pel fet que hi ha més persones disposades a oferir-les. Per altra banda, aquestes ofertes també arribaran a més gent.

Els requeriments que les ofertes havien de complir perquè les consideréssim vàlides per a l'aplicació eren clars: havien de ser ofertes vigents, situades a la ciutat de Barcelona preferiblement, o a nivell de Catalunya, que fos possible obtenir la seva localització exacta i que les fonts fossin fiables.

Vam considerar moltes fonts, finalment vam optar per obtenir les ofertes d'una sèrie de webs. Aquestes són:

- Groupon
- Groupalia
- Restalo
- El Tenedor

Groupon

Groupon és una web on s'ofereixen diversos cupons que ofereixen una gran diversitat de serveis o productes a un preu reduït. Quan un usuari es registra, aquest rep ofertes a la seva adreça de correu electrònic que s'adapten als seus interessos . Evidentment, aquestes ofertes també es poden consultar al mateix web. Generalment ofereixen cupons que si hom vol utilitzar-los, els ha de comprar.

Aquesta part del web no interessava, ja que oferir comprar cupons des d'una aplicació mòbil, implicava oferir una compra des d'un terminal mòbil, cosa que pot portar, en primer lloc, molta desconfiança a l'usuari, i en segon lloc, un problema de seguretat important, ja que fer un pagament és una acció crítica, i vam veure que era un risc que no volíem assumir.

Per altra banda, està molt allunyat de la filosofia que l'aplicació seguirà, aquesta simplement oferirà una informació a l'usuari relacionada amb les ofertes que té al seu voltant. Si l'usuari vol consumir una d'aquestes ofertes, haurà d'anar al seu lloc d'origen i fer la compra.

Per sort, Groupon també té un catàleg (tot i que molt més limitat) d'ofertes que s'adapten als nostres principis. Aquestes són les ofertes que s'han recopilat de Groupon.

Per a fer això, s'ha implementat un Web Crawler[1] en php[2], utilitzant Symfony 2 [3](Framework [4]php que ha fet les coses més fàcils en aquesta fase) que accedia a les URL on es trobaven aquestes ofertes, les consultava i les guardava a la base de dades.

Aquest crawler s'ha implementat utilitzant el Crawler "Goutte" com a base. Aquest crawler forma part d'un mòdul ja desenvolupat del Framework, les funcionalitats del qual s'adaptaven totalment a les nostres necessitats. S'ha utilitzat i adaptat en cada cas que ha estat necessari per a poder obtenir les ofertes de diverses webs.

Aquest procés d'obtenció de dades es fa periòdicament, per a acumular la màxima quantitat d'ofertes possible. Quan s'executa, la primera cosa que es fa és consultar la base de dades, i esborrar totes les ofertes que hagin caducat, ja que considerem que mostrar a l'usuari ofertes que ja no són vàlides és una raó per la qual els usuaris deixarien d'utilitzar l'aplicació.

[1] **Web Crawler**: Programa que inspecciona les pàgines web de forma metòdica i automatitzada.
[2] **Php:** Llenguatge de programació interpretat, dissenyat originalment per a crear pàgines web dinàmiques. S'utilitza principalment per a la interpretació del costat del servidor (server-side scripting).
[3] **Symfony**: És un complet Framework php, dissenyat per a optimitzar el desenvolupament d'aplicacions web, que proporciona diverses eines i classes encaminades a reduir el temps de desenvolupament d'una aplicació web complexa.
[4] **Framework:** Estructura conceptual i tecnològica de suport definit, normalment mitjançant artefactes o mòduls de software concrets, amb base als quals, un altre projecte de software pot ser més fàcilment organitzat i desenvolupat

Com a mitjana, es tenen unes vint ofertes de la ciutat de Barcelona diàriament.

Groupalia

El funcionament d'aquesta web és idèntic al de Groupon, ofereixen gran quantitat de descomptes i cupons, els quals s'han de comprar per a poder utilitzar. Per sort, també ofereixen ofertes convencionals, que podem utilitzar a l'aplicació.

El seu catàleg d'ofertes és extens, però la gran majoria són cupons. Per tant, la part en la qual ens centrarem serà la part més minoritària de les seves ofertes, aquelles que no necessiten la compra del cupó.

S'ha utilitzat la mateixa metodologia que amb Groupon, s'ha adaptat el Web Crawler perquè la web de la qual s'extregui la informació sigui Groupalia, en lloc de Groupon.

D'aquesta font, es tenen unes quinze ofertes vigents de la ciutat de Barcelona.

Restalo

Aquest cas és totalment diferent, ja que això no s'ha fet mitjançant el crawler del què ja hem parlat, sinó que s'ha arribat a un acord comercial amb l'empresa, i se'ns ha donat accés a la seva API[5], per a consultar la informació que ens faci falta.

Es tracta d'una web on s'ofereixen reserves de restaurants de més de quaranta províncies espanyoles. Donen la possibilitat de reservar des del seu web, a més d'oferir gran quantitat de reserves amb oferta inclosa.

Aproximadament uns 8000 restaurants catalans tenen compte de Restalo, i donen la possibilitat de reservar mitjançant aquesta web.

Aquesta font ha estat protagonista d'aquesta fase prèvia del projecte, ja que als estudis del Twitter es mostraven aquestes ofertes als usuaris, per a estudiar el seu comportament (veure l'apartat "Promoció d'ofertes al Twitter").

Hem considerat que els continguts aportats per aquest servei, eren molt atractius per a la nostra aplicació, i que tenien molt de sentit, ja que, donat que el que ofereixen en aquest web, són reserves, i no s'ha de pagar res, a un usuari no li generarà desconfiança, sinó que li resultarà una proposta molt interessant.

[5] **API (Application Programming Interface)***:* Conjunt de funcions i procediments que ofereix certa llibreria per a ser utilitzada per un altre software com a capa d'abstracció. Moltes pàgines web, ofereixen una API pública, perquè altres desenvolupadors utilitzin la informació que tenen emmagatzemada.

Reservar des d'un dispositiu mòbil pot tenir molt de sentit, ja que si a hora de dinar, tens dubtes d'on dinar, igual amb l'aplicació, se t'aclarirà el dubte i t'assegurarás taula mitjançant la reserva. A més, la seva web té versió mòbil, totalment adaptada als dispositius actuals, punt crucial, ja que d'aquesta manera, faciliten la navegació des dels mateixos.

El Tenedor

Aquesta web forma part de la cadena de restaurants El Tenedor, cadena que a Espanya té uns 4500 restaurants per totes les comunitats autònomes. En la web s'anuncien els diversos descomptes que hi ha als restaurants de la cadena.

Per a agafar aquestes ofertes, s'ha utilitzat la mateixa metodologia que amb Groupon i Groupalia, s'ha adaptat el crawler perquè obtingui la informació d'aquesta web, i l'emmagatzemi a la base de dades. Evidentment, aquestes ofertes es van agafant periòdicament, i es van esborrant les que ja hagin caducat.

Aquestes són les fonts que s'han explotat per a tenir una quantitat d'ofertes suficient per a començar a testejar l'aplicació en condicions. Evidentment, això és un aspecte que s'haurà d'ampliar en un futur, per tal d'oferir una gran quantitat d'ofertes de diverses fonts als usuaris.

Estudi de les zones on més s'utilitzen els smartphones

En aquesta fase de l'estudi previ, l'objectiu que es vol assolir, és descobrir quines són les zones on els usuaris tenen tendència a utilitzar els seus smartphones. Això s'ha estudiat a nivell de Catalunya. El que es vol veure és si hi ha zones on la gran majoria utilitza els smartphones, i veure si aquestes zones resulten ser zones comercials. De ser així, tenir una aplicació amb les ofertes del seu voltant, definitivament, seria una idea més que interessant, ja que demostraríem que el públic objectiu, és a les zones que ens interessen, utilitzant els dispositius que ens interessen.

Per a realitzar aquest estudi, s'utilitzaran principalment dues xarxes socials que estan molt de moda a l'actualitat: Twitter i Foursquare. Abans de res, s'explicarà una mica què és cadascuna, per a posar-nos en context, i veure el sentit del que s'està fent.

Twitter

Twitter és una xarxa social [6]orientada al microblogging[7]. És a dir, els usuaris tenen la possibilitat d'escriure al seu perfil, però només missatges curts (de 140 caràcters com a

[6] **Xarxa Social**: Estructures socials compostes per grups de persones, les quals estan connectades entre elles per diferents tipus de relacions, com puguin ser interessos comuns, amistat, parentesc o coneixements comuns. Clars exemples de xarxes socials: Facebook, Twitter, Youtube, Myspace, Linkedin, Instagram...

màxim). Es va crear al març del 2006, i actualment s'estima que té més de 200 milions d'usuaris.

Els usuaris es poden seguir entre ells, enviar-se missatges, i una sèrie d'opcions més que han fet que aquesta xarxa social sigui de les més utilitzades a l'actualitat.

Un detall diferenciador de Twitter és que un usuari pot seguir-ne a un altre, sense que aquest altre el segueixi. És a dir, el concepte de "seguidor" (al Twitter, això es coneix com ser "follower") és diferent al més típic, d'"amic", ja que en aquest cas, només seguiràs a les persones que t'interessin, i només et seguiran a tu, les persones a les quals interessis.

Foursquare

Foursquare és una xarxa social orientada a la geolocalització. La gent que té instal·lada l'aplicació mòbil pot veure els llocs que estan enregistrats del seu voltant, i en el cas que estigui a un d'ells, fer el que ells anomenen "check-in", que consisteix simplement a dir que ets en aquell lloc. Permet als usuaris enregistrar nous llocs, als quals es pot assignar una categoria i una descripció, entre d'altres característiques configurables.

Té tot un sistema del que es coneix com "achievements[8]" en anglès, que consisteix a premiar els usuaris pel fet de ser els que més check-ins han fet a un lloc en concret (et diuen que ets "el batlle" del lloc) o per altres coses com pot ser fer check-ins en molts de llocs diferents.

Actualment tenen uns cinc milions d'usuaris, i compten amb aplicacions per a la gran majoria dels sistemes operatius mòbils actuals.

En aquest punt, que ja sabem què són el Twitter i el Foursquare, podem avançar i comentar com s'han utilitzat en aquesta fase.

Foursquare ofereix l'opció de connectar-se amb Twitter, cosa que consisteix en el fet que quan un usuari fa un check-in, al Twitter, automàticament, se li genera un Tweet amb aquesta estructura:

[Missatge de l'usuari] (@ [Lloc del check-in]) [URL de Foursquare]

D'aquesta manera, té un Tweet, escrit al seu perfil, que expressa que és a un lloc en concret, enregistrat al Foursquare.

---

[7] **Microblogging:** Servei que permet als usuaris enviar i publicar missatges breus, generalment de text pla únicament.

[8] **Sistema de Achievements**: Sistema que implementen moltes xarxes socials i jocs actuals, on es premia als usuaris segons les accions que fan amb el programa en concret. En el món dels videojocs es sol conèixer com "Trofeus" i en les xarxes socials es pot veure fàcilment amb el sistema de medalles de Foursquare, o en el sistema dels jocs de Facebook, on et recompensen amb punts del mateix joc, si has complert una sèrie d'objectius.

Doncs per veure en quin tipus d'establiments, i sobretot, en quines zones de Catalunya s'utilitzaven més els smartphones, el que s'ha fet ha estat capturar periòdicament els Tweets geolocalitzats, que corresponen a check-ins de Foursquare, i s'han representat en un mapa, per a poder veure les dades clarament. Per fer això, s'ha utilitzat l'API de Twitter, que permet fer cerques explicitant les coordenades centrals de la cerca i un radi, de tal manera que dins la circumferència de cerca, només hi teníem Catalunya.

Com he esmentat, això era una tasca que es tenia automatitzada[9], i de mitjana, diàriament, s'han capturat entre 250 i 400 Tweets (depenent sobretot del dia de la setmana).

Representació de la informació

Per a representar tota la informació de què disposem, s'ha utilitzat l'API de Google Maps, que ens permet marcar diferents posicions geogràfiques amb Markers[10], i en aquests tenir la informació que ens convingui.

Com tenim emmagatzemats a la base de dades una gran quantitat de Tweets (al moment de redacció d'aquest text, en tenim més de 8000), veure tots els Tweets amb Markers dins un mapa de Catalunya és molt caòtic, i no en podem treure cap tipus de conclusió.

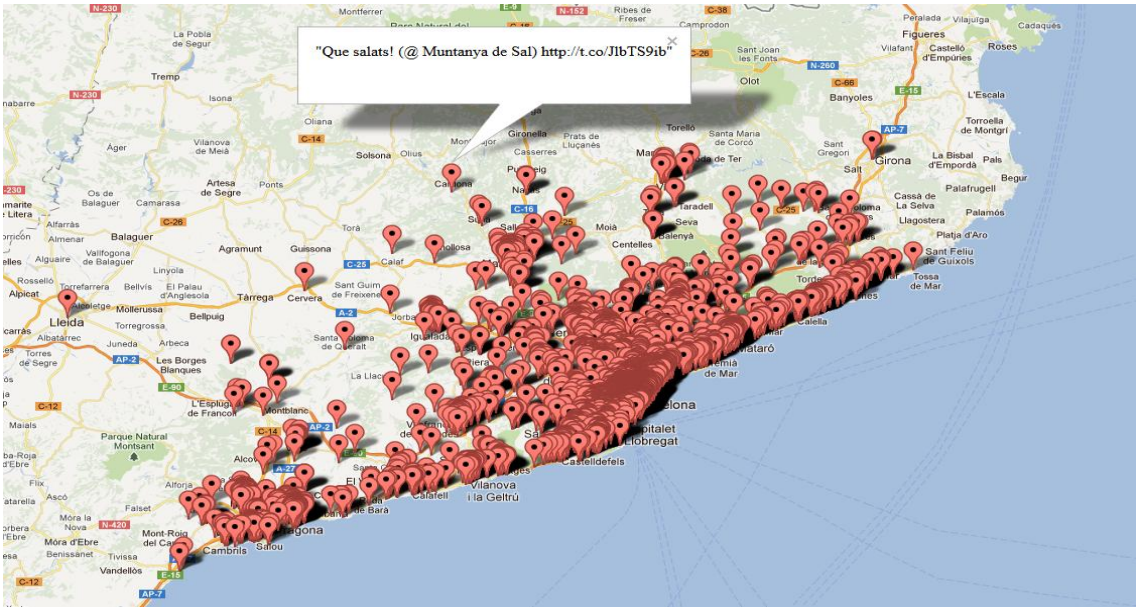
Per aquesta raó, s'han implementat dos tipus de filtratge, que no són excloents (és a dir, es poden aplicar els dos alhora) :

- **Per data**: Se li pot indicar a la visualització del mapa, quin rang de dates es vol mostrar. De tal manera que si només vols veure representats els Tweets d'una data en concret o d'una setmana, o en general, de qualsevol rang de dates que vulguis, es pot fer d'una manera tan senzilla com especificar dues dates (inici i fi del rang), i només se't mostraran les dades corresponents.

- **Per agrupació de Markers**: Aquest filtre el que indica és que en lloc de mostrar Markers individualment, el que es fa és mostrar el que s'anomenen "clústers" de Markers, que bàsicament l'únic que fan és agrupar els Markers, per a veure les zones on n'hi ha més d'acumulats de manera senzilla.

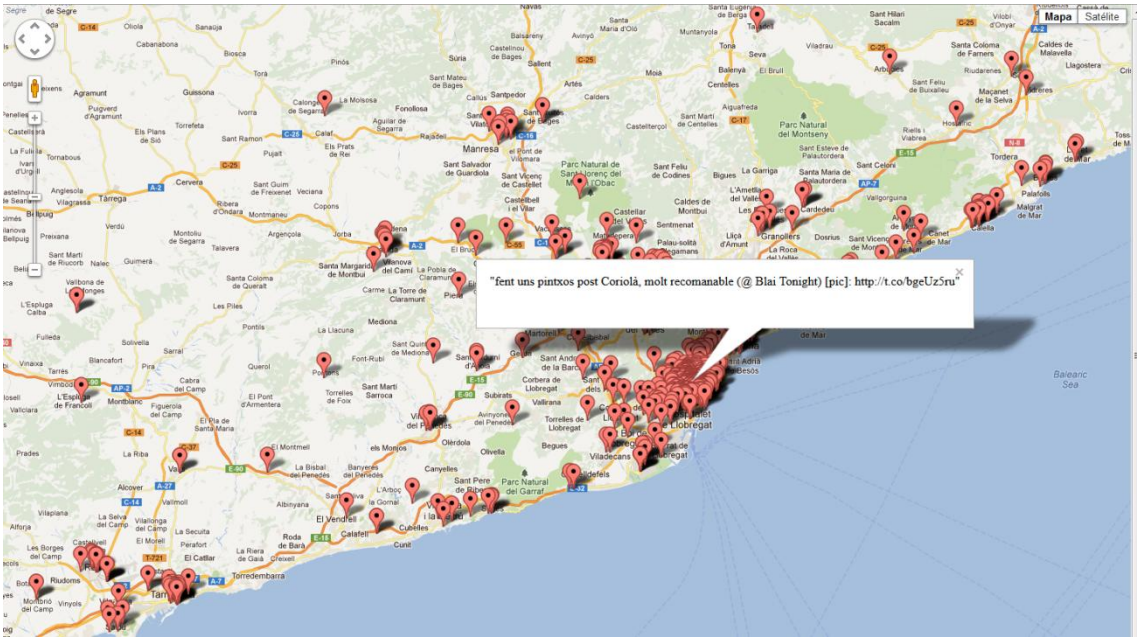
[9] Això s'ha fet programant la taula de processos (crontab) del servidor. Com al servidor treballem amb una distribució de linux, això ha estat tan senzill com editar un fitxer de configuració del sistema.
[10] **Marker**: En el context de l'API de google maps, un Marker, o marcador, és un indicador que es col·loca en una posició concreta del mapa, per a representar algun tipus d'informació. En el nostre cas, cada Marker mostra la posició geogràfica d'un Tweet, i si cliques damunt, pots veure el Tweet en qüestió.

Per a entendre més fàcilment aquests filtres, es mostraran una sèrie d'exemples on quedarà tot més clar:



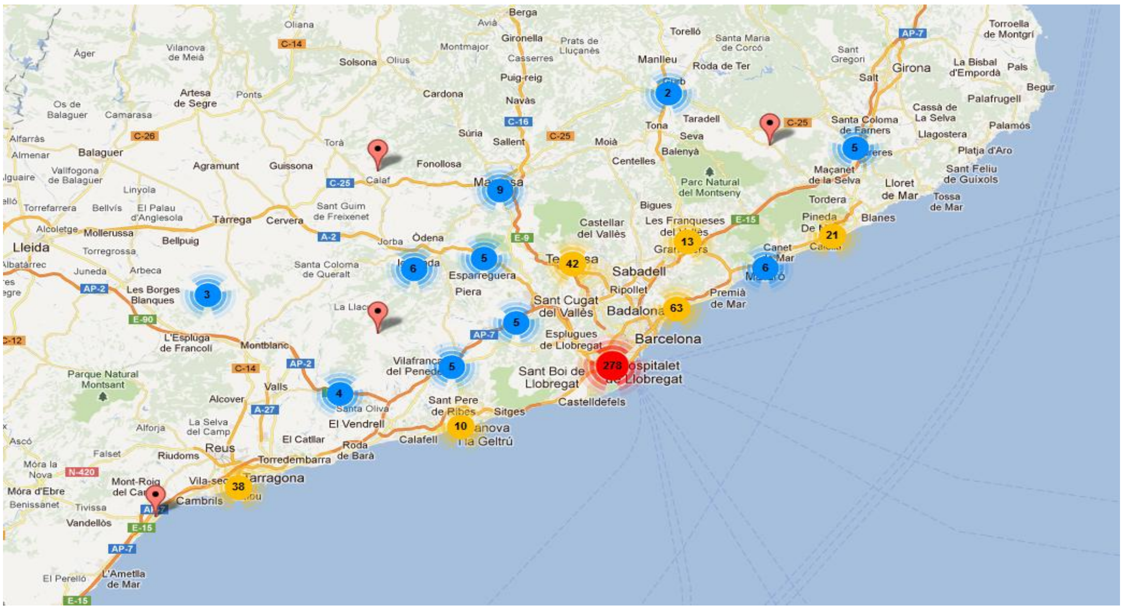
En aquesta captura, podem veure bàsicament com queden tots els Markers junts, sense cap tipus de filtratge. Com hem comentat abans, amb aquesta visualització, és complicat distingir que és el que està passant.

Ara mostro una captura de pantalla, amb una visualització del mateix estil, però que només es correspon a un dia en concret (en aquest cas, el 24 de març):



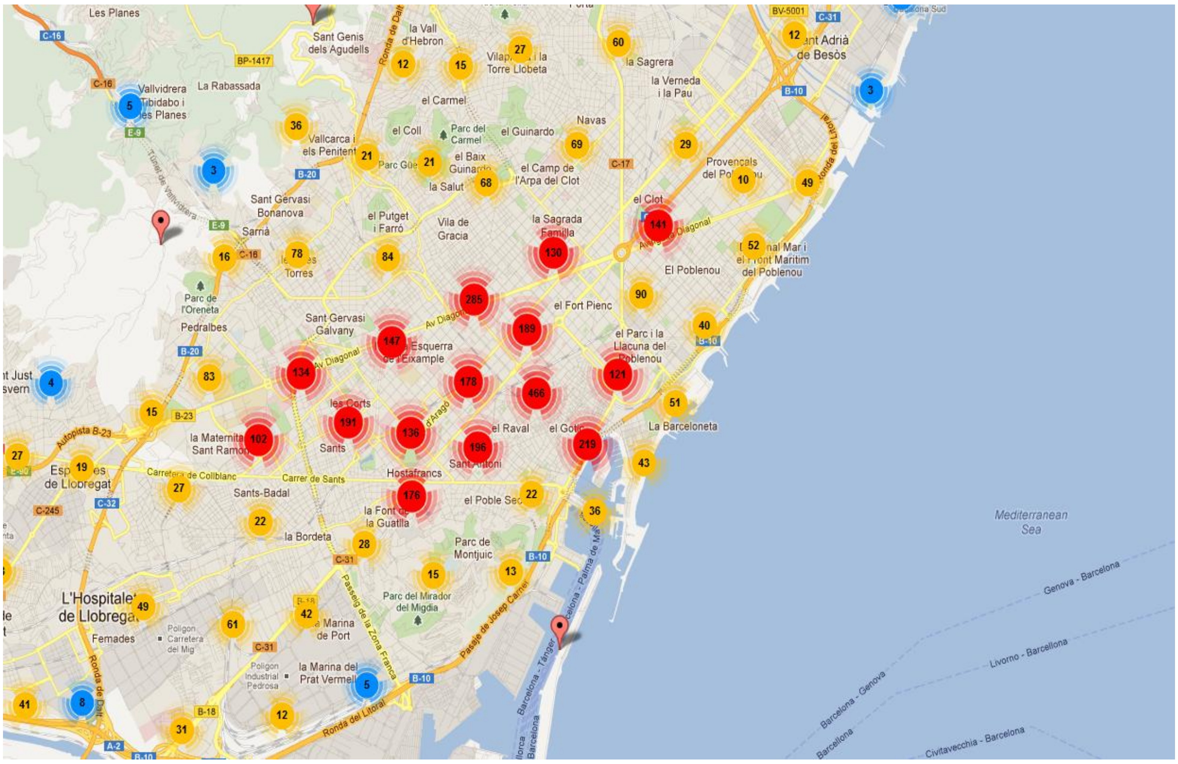
Com podem veure, tot i que la informació és evidentment menys abundant, segueix essent poc clara la representació.

Si ara, per a aquesta mateixa data, apliquem un l'altre filtre (agrupació de Markers), obtindrem aquest resultat:



D'aquesta manera, ja es veu més clarament on hi ha més activitat. Com era previsible, la quantitat més gran de Tweets prové de la ciutat de Barcelona. Amb aquest filtre, si fem zoom, podem veure com les rodones es van desglossant per barris.

Ara que ja sabem que la font més gran d'activitat és a la ciutat de Barcelona, mostraré, amb agrupació de Markers, com es reparteixen els Tweets per barris dins la ciutat. Remarquem que no filtraré per data, sinó que es mostraran els valors corresponents a totes les dades que a dia d'avui tenim a la nostra base de dades.



Al mapa anterior podem observar que amb 466 Markers, la zona on més Tweets hi ha concentrats és la de plaça Catalunya- El Raval. Analitzant aquestes dades arribem a la conclusió que és per aquesta zona per on la gent fa més check-ins habitualment. També veiem que hi ha activitat per plaça Espanya, per Sants, i en general, per tots els barris del voltant de la Diagonal de Barcelona. Com ja es podia preveure, els check-ins es fan generalment en zones comercials de la ciutat.

Aquesta conclusió ens fa veure que l'aplicació que programarem pot tenir un gran nombre d'usuaris, si aconseguim una gran quantitat d'ofertes que estiguin localitzades en aquests barris de la ciutat de Barcelona.

Un altre focus d'activitat del qual no hem parlat, ni l'hem mostrat al mapa anterior, és l'aeroport, on hi ha uns 250 Tweets, així que també podem veure que allà és on la gent generalment fa Check-ins. Això no ho valorem, ja que la gran majoria de les persones que passa per l'aeroport de Barcelona, no és gent de la zona, és gent de fora de Barcelona, que segurament no estigui interessada en una aplicació destinada a mostrar les ofertes de la ciutat, pel fet que la seva estada, a la ciutat, en general, serà curta.

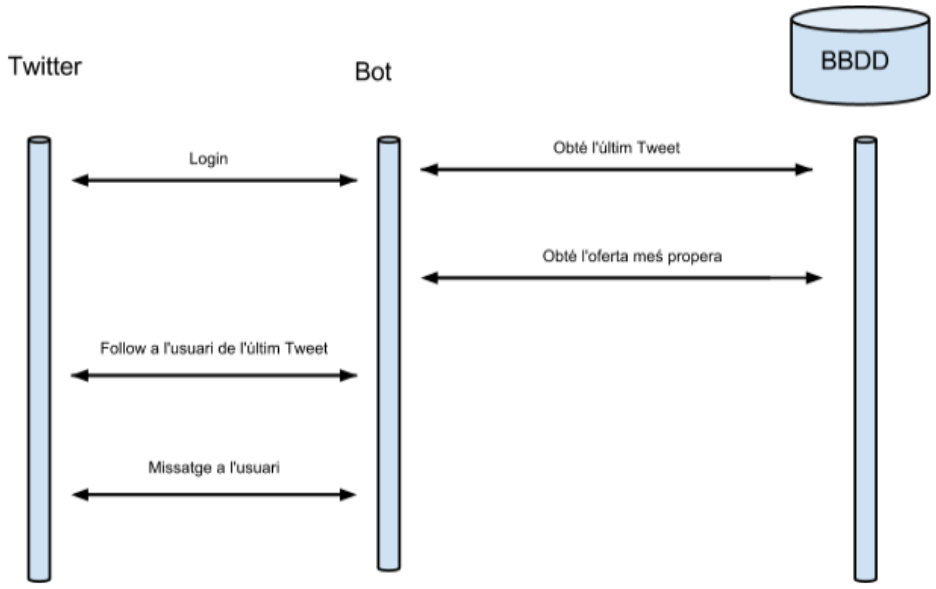
Promoció d'ofertes per Twitter

Descripció General del procés

Després de la recopilació d'ofertes i de Tweets geolocalitzats corresponents a check-ins de Foursquare, el que s'ha fet ha estat relacionar aquests Tweets, amb les ofertes recopilades.

Per a fer això, s'ha programat un bot al Twitter que assignés als usuaris que han generat els Tweets capturats l'oferta més propera a la posició on s'havia publicat el Tweet. L'objectiu d'això era enregistrar el comportament dels usuaris, analitzant com reaccionaven davant aquestes ofertes.

Per a comprendre millor el flux d'execució del bot, amb un petit diagrama, s'entendrà millor:



Com es pot veure al diagrama, primer s'obté l'últim Tweet capturat. Amb aquesta informació, obtenim tant la seva localització, com el seu autor. Amb la seva localització, busquem quina oferta és la que té més a prop. Per altra banda, el bot fa login al Twitter, es fa seguidor de l'autor del Tweet, i, finalment, li envia un missatge. El missatge, en concret, es tracta d'una menció[11], que té aquesta estructura:

@[Nom Usuari] [Text Descriptiu de l'oferta] [URL de la Oferta]

D'aquesta manera, l'usuari rep una menció del perfil creat per a ser el bot.
Donat que ara tenim una visió general del procés, ara aprofundirem en cada un dels passos que hem esmentat.

[11] **Menció**, al Twitter és un Tweet que va dirigit a un usuari o usuaris en concret, que té com a estructura: @[user1]...@[userN] [Missatge]. Aquests missatges es mostraran en els perfils dels usuaris esmentats.

Com s'ha programat?

El bot no és més que una classe escrita en php, que utilitza l'API de Twitter per a interactuar amb aquesta xarxa social. Les crides que es fan, es fan mitjançant la llibreria CURL [12] de php. Aquestes crides es comuniquen amb unes entitats del Twitter, que ells mateixos anomenen "aplicacions[13]", que són les responsables de gestionar la comunicació de programes escrits en diversos llenguatges de programació (en el nostre cas, php) amb el Twitter. S'indica que aquesta aplicació es pot comunicar amb el perfil del Bot (cosa que es pot fer des de les opcions del perfil del bot al Twitter), i ja funciona aquest procés automàtic.

Aquest procés, com ja he comentat, s'ha automatitzat, perquè entre les 11 i la 1 del matí, i les 7 i les 10 de la tarda, vagi fent mencions als usuaris de Twitter.
Això té una raó de ser, i és principalment que la informació que s'envia, és l'obtinguda de Restalo (la web de reserves en Restaurants de la qual hem parlat abans) . D'aquesta manera, enviem la informació en horaris en què la gent té tendència a voler anar a dinar o a sopar fora de casa, i en definitiva, horaris en què aquesta informació pot ser útil.

El link amb l'oferta que s'envia a l'usuari, el redirecciona a la web de reserva del Restaurant que s'ha trobat més a prop de la situació de l'usuari al seu últim Tweet capturat, enviant d'aquesta manera un restaurant proper a un lloc on l'usuari ha estat abans, amb la qual cosa, possiblement, aconseguim cridar-li l'atenció.

Per a fer més potents aquestes mencions, no només es diu a l'usuari que hi ha un restaurant amb oferta, i se li envia un link, sinó que la menció és una mica més complexa. Ho veurem perfectament amb un exemple real:

En aquest cas podem observar com a l'usuari se li ha ofert una reserva al restaurant "Thai Lounge", que s'ha calculat que és a 54 metres del "Restaurant Habaluc", el lloc on l'usuari ha fet "check-in" en el seu últim Tweet capturat. Per últim se li envia un link amb l'oferta en concret.

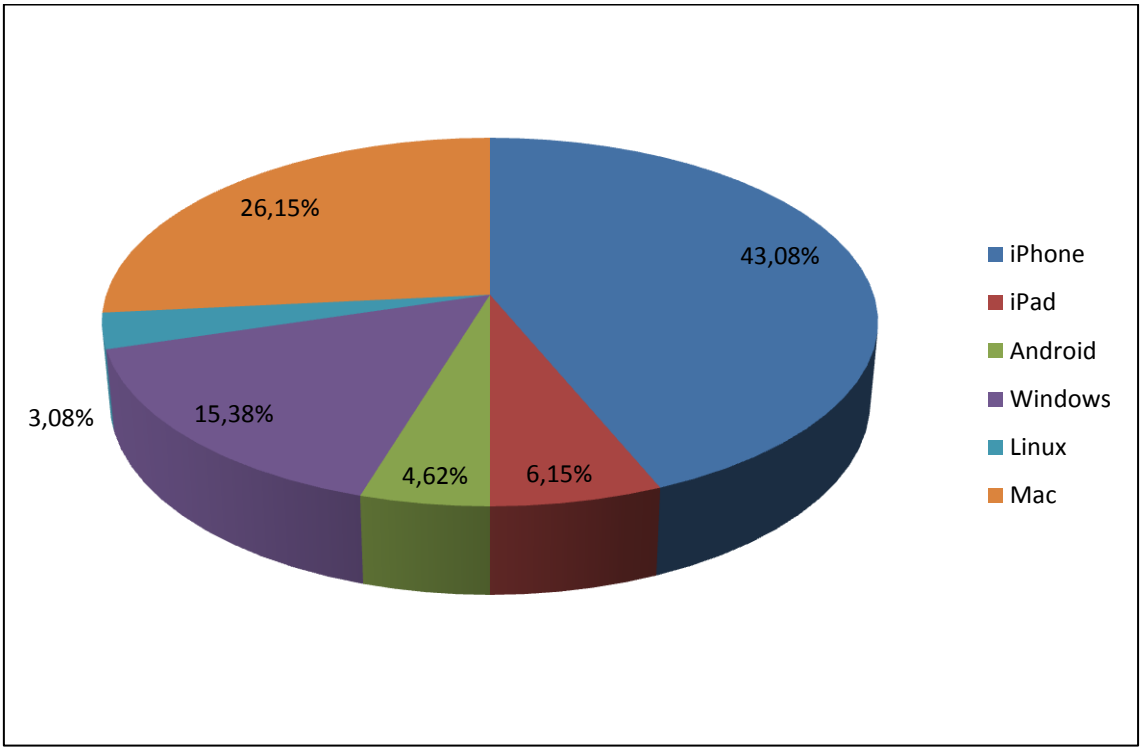
El que s'intenta aconseguir amb això és que l'usuari vegi una informació referent a un lloc conegut per ell, i que amb això creem una confiança, que faci que cliqui al link, per a veure què és el que se li està oferint.

[12] **Llibreria Curl**: Llibreria en aquest cas de php, que simula el comportament d'un navegador web.
[13] **Aplicacions de Twitter:** Al Twitter, són les encarregades de connectar amb el perfil d'un usuari, de tal manera que si es vol automatitzar accions, s'haurà d'enregistrar una aplicació vinculada amb el perfil al qual es vulgui actuar.

Una vegada s'ha fet la menció, s'estudia el comportament de l'usuari. Per a fer això, el que es fa és enganyar una mica l'usuari, que pensa que se l'ha enviat directament a la web de l'oferta, mentre que realment el procés és una mica diferent.

Primer el redireccionem a una URL coneguda del nostre servidor, on s'emmagatzema una sèrie de característiques de l'usuari que ha clicat, com són la seva IP[14], el User-Agent[15] del seu navegador i el temps que ha transcorregut entre la menció i el click de l'usuari. Posteriorment, se'l redirecciona a la URL del restaurant. Gràcies a aquest procés transparent a l'usuari, hem aconseguit caracteritzar el comportament dels usuaris.

A continuació, mostrem un gràfic i unes estadístiques que representen algunes de les conclusions que hem extret d'aquest procés, primer mostrarem els tipus de dispositius des dels quals els usuaris han fet click als links de les ofertes que oferíem al Twitter:



Podem destacar que Apple té la majoria absoluta, ja que al camp dels dispositius mòbils, guanyen amb l'Iphone i amb els sistemes operatius d'escriptori, també, amb el MacOs.

[14] **Direcció IP**: Etiqueta numèrica que identifica de manera lògica i jeràrquica una interfície d'un dispositiu dins d'una xarxa IP.
[15] **User-Agent**: En aquest cas, em refereixo al camp de les capçaleres HTTP que s'envien quan es fa una petició. Al camp l'User-Agent, tenim informació de l'aplicació, la versió, el sistema operatiu i l'idioma de l'usuari que ha fet la petició.

Cal destacar també el fet de que només un 4.62% d'usuaris fes click des d'un terminal amb Android, podem anar veient que els usuaris d'Apple catalans són els més receptius quant a ofertes.

També hem de destacar el tant per cent del total dels usuaris que han fet click. Després de fer més de 800 mencions, s'ha vist que aproximadament **un 25% dels usuaris mencionats, feia click a l'oferta**. Un tant per cent més alt del que inicialment s'havia previst (s'estimava que clicarien entre un 10 i un 15%).

La mostra de Tweets totals que s'han arribat a capturar periòdicament, a la data actual, era d'uns 9200, capturant aproximadament 300 per dia.

Per últim, també s'ha calculat el temps que passava entre la menció i els clicks, i de mitjana, **els usuaris trigaven unes 12 hores a clicar damunt de l'enllaç**. Això ens fa veure que la gent es connecta periòdicament al Twitter, però no hi està contínuament. Per a fer aquest càlcul, enregistràvem l'hora en què s'havia fet la menció i l'hora del click. Per a obtenir un valor mitjà del temps que passava entre un event i l'altre de manera fiable, s'han eliminat els casos més extrems, ja que hi havia un parell de casos on el temps de click era d'unes dues setmanes, corresponent a usuaris que es connecten molt poc al Twitter, i que desviava molt el valor real de la mitjana. També s'han eliminat els clicks instantanis que s'havien produït en un parell d'ocasions també, i que hem considerat massa coincidència com per a tenir-los en compte.

Problemes que han sorgit

Durant aquest procés d'estudi del comportament dels usuaris, han sorgit una sèrie de problemes que han complicat una mica el procés.

El primer inconvenient que ha sorgit ha estat el del baneig[16] d'aquests perfils d'usuari que s'han creat per al bot. Twitter té una política que restringeix l'automatització d'enviament de missatges, fixant unes quotes màximes que no expliciten. Per a trobar aquests màxims, s'ha anat provant, baixant la freqüència de les mencions, fins que s'ha arribat a un punt en el qual Twitter no banejava als Bots. Inicialment es feien les mencions de 4 en 4, a les mateixes hores sempre, i de manera molt freqüent. Es va descobrir que la periodicitat constant dels missatges i les mencions de membres de Twitter, era una característica del bot que no agradava als responsables d'aquesta xarxa social.

Per a corregir això, es va baixar la freqüència dels missatges (a uns 30 al dia), es va anar canviant les hores on s'executava i es van intercalar missatges sense menció. Un altre aspecte amb el qual també vam haver de vigilar, va ser amb el fet que, després que ens banegessin el primer bot, al segon, el van banejar quan no havia publicat ni 10 Tweets.

[16] **Banejar:** Expulsar, o eliminar el dret d'accedir a un determinat servei per una violació de les normes d'utilització del servei en concret.

Això es va solucionar enregistrant-nos des d'un Web Proxy[17], de tal manera que Twitter no detectés la mateixa IP. Això és un aspecte que Twitter controla molt bé en la versió per a navegadors de la seva web, però que a dia d'avui, no controla bé en la seva versió mòbil.

Per acabar d'arrodonir aquestes mesures de precaució, s'han creat dos perfils de Twitter (dos bots), i s'han anat alternant, de tal manera que cada dos dies aproximadament, es canviava de perfil, perquè no es detectés tan clarament una periodicitat en les accions a cadascun dels perfils.

Així s'ha aconseguit fer aquestes mencions sense que la gent de Twitter ho considerés una violació de les condicions d'ús.

Un altre problema amb el qual ens hem trobat, ha estat en l'emmagatzematge dels clicks que feien els usuaris sobre els links que enviàvem als seus perfils periòdicament. Com hem dit, quan fèiem la redirecció, enregistràvem el click, i desàvem informació sobre la procedència d'aquest click. Ens trobàrem que hi havia més clicks que mencions (en un punt, teníem que amb 100 mencions,s'havien enregistrat 140 clicks). Inicialment vam celebrar aquesta estadística, ja que era totalment inesperada i que ens feia pensar que tothom estava interessat en la informació que oferíem.

Vam veure que ens havíem equivocat quan vam detectar moltes ip's repetides, i User Agents que no coneixíem[18].

Després d'investigar una mica, vam descobrir que la gran majoria d'aquests clicks eren generats per processos automatitzats que corren per Twitter, que seguien els links que oferíem als usuaris. Això era possible a causa del fet que la redirecció la fèiem al servidor (en php) i els bots podien seguir les URL[19]. Per a solucionar això, la redirecció la vam fer al navegador de l'usuari (en javascript)[20].
D'aquesta manera, vam obtenir unes dades molt més precises, on veiem que un 25% de les persones realment feien click, valor que encaixava molt millor amb les expectatives inicials.

Ens va sobtar aquest fet, ja que la gran majoria de les visites eren de processos automatitzats dins del Twitter, cosa que personalment pensava que no era tan freqüent.

[17] **Web Proxy**: Es tracta d'un tipus d'aplicacions web que fan d'intermediari en les peticions que fa l'usuari. D'aquesta manera, s'aconsegueix que la IP de l'usuari, per al receptor, passi a ser la del Web Proxy.
[18] S'han fet diversos estudis, que indiquen que un 51% del trànsit que hi ha actualment a internet, el generen processos automatitzats.
[19] Això passa perquè les redireccions al servidor es fan mitjançant una petició http a un altre domini, cosa que servirà per redireccionar tant a usuaris humans, com a bots.
[20] Amb javascript, només es redireccionaran a usuaris humans, ja que aquesta redirecció es fa a nivell de navegador del client, cosa amb la qual no compten els bots.

Després de veure amb una mica de detall els estudis previs, podem treure una sèrie de conclusions que ens serviran per a enfocar de manera més precisa el desenvolupament de l'aplicació.

Conclusions Finals Estudis Previs

- Els usuaris catalans de Twitter, que també utilitzen Foursquare, tenen curiositat per les ofertes que els hem enviat al Twitter.
- Les zones comercials i l'aeroport són els llocs on la utilització dels smartphones és més important
- Els usuaris als quals s'ha fet una menció i cliquen damunt de l'enllaç, triguen unes 12 hores a fer-ho.
- Els usuaris que tenen més tendència a interessar-se per ofertes són els de dispositius Apple.
- Hi ha molts usuaris d'smartphones a Catalunya.
- Es poden trobar de manera senzilla ofertes geolocalitzades periòdicament actualitzades a la zona de Catalunya (sobretot a Barcelona ciutat).
- Una gran quantitat del trànsit present al Twitter és un procés automatitzat.

Estudi Tecnologies

Visió general

Per al desenvolupament d'aquesta aplicació, s'han de triar les tecnologies més adequades, per tal que la programació sigui còmoda, es puguin assolir els objectius marcats i es pugui fer una aplicació amb un bon rendiment.

Considerem el rendiment de l'aplicació com el punt més important, ja que els usuaris són molt exigents en aquest aspecte. Una aplicació lenta, o poc fluïda, serà una aplicació que es desinstal·larà de manera pràcticament instantània. Un altre aspecte important és l'aspecte visual, ja que els usuaris estan acostumats a una interfície molt cuidada i no toleraran una interfície poc clara, o poc atractiva estèticament.

Dins de tot l'ample ventall de tecnologies mòbils disponibles, estudiarem principalment dos tipus de tecnologies. Per una banda, hi ha les tecnologies destinades a fer aplicacions web, que s'aprofiten de la capacitat dels terminals mòbils d'interpretar Html5 i javascript[21], i permeten, amb l'ajuda de serveis de compilació d'aquestes tecnologies, programar una vegada l'aplicació, i compilar-la per a la gran majoria dels dispositius mòbils actuals.

Per altra banda, també considerarem la possibilitat de centrar-nos en un tipus de dispositius en concret, i fer una aplicació específica.

Les dues tipologies de tecnologies presenten uns pros i uns contres. En aquest apartat, es farà un estudi de la viabilitat, els avantatges i els inconvenients que presenta cada opció. Finalment, després d'aquest estudi, s'escolliran les tecnologies que s'utilitzaran per a desenvolupar l'aplicació.

Abans de posar-nos en aquest aspecte, s'ha de comentar les tecnologies que s'utilitzaran al costat del servidor. Com s'ha vist als estudis previs, al costat del servidor, el php és el llenguatge amb el qual més facilitat tinc per a desenvolupar. Donada aquesta circumstància, l'aprofitaré i el codi de la part del servidor, estarà programat en php. El servidor es connectarà amb una base de dades Mysql[22], tecnologia amb la qual també tinc experiència. Ambdues tecnologies tenen un rendiment bo, i s'adapten perfectament als interessos del projecte. Tot aquest aspecte es podrà veure clarament quan es presenti més endavant, a l'apartat de la implementació.

[21] **Javascript**: Llenguatge de programació interpretat, que s'utilitza principalment al costat del client, implementat com a part del navegador web de l'usuari. Permet la creació de pàgines web dinàmiques amb interfícies molt elaborades. S'utilitza també en molts altres contexts.

[22] **Mysql:** Sistema gestor de bases de dades relacional molt popular actualment, compta amb controladors específics per a gran quantitat de llenguatges de programació, com php, C, C++, C#, Pascal, Java, Lisp, Perl, Python...

Tecnologies mòbils que cal estudiar

Com hem esmentat abans, primer estudiarem les tecnologies orientades a programar aplicacions multiplataforma.

En concret, estudiarem jquery mobile i Sencha touch, les quals combinades amb el compilador d'aplicacions, Phonegap, permeten programar aplicacions mòbils multiplataforma. Abans de res, explicarem phonegap:

Phonegap

Phonegap està format per dos serveis principals. El primer ja l'hem esmentat i és el de compilació d'aplicacions. El que fa és agafar codi escrit en Html5, javascript i CSS3[23] i el transforma en codi compatible amb els principals dispositius mòbils actuals. Actualment compila per iOS, Android, Blackberry, WebOs[24] i Symbian[25], i estan treballant en la compilació d'aplicacions per Windows Phone, Bada[26], i altres sistemes operatius mòbils.

Per altra banda, Phonegap ofereix un SDK multiplataforma que el que fa és permetre, de manera transparent al programador, accedir des de javascript, a les funcions pròpies del telèfon (cosa que d'altra manera, seria impossible), com poden ser el GPS, la càmera de fotos, o la vibració. D'aquesta manera, el programador pot accedir a aquestes funcions, sense preocupar-se de com es fa en cada dispositiu en concret.

La combinació dels dos serveis permet desenvolupar aplicacions de manera senzilla, arribant de manera ràpida als principals sistemes operatius mòbils actuals.

Phonegap és un servei desenvolupat per una companyia anomenada Nitobi (que ha estat comprada per Adobe), nascuda al 2009 i que actualment té gran popularitat en el món del desenvolupament d'aplicacions mòbils.

Aquest servei és molt útil, però s'ha de combinar amb algun Framework que ajudi a estructurar, optimitzar i donar un aspecte d'aplicació mòbil al javascript que conté la lògica de l'aplicació.

---

[23] **CSS**: Llenguatge utilitzat per a definir l'aspecte d'un document escrit en HTML o XML.

[24] **WebOs:** Sistema operatiu mòbil, inicialment desenvolupat per Palm Inc, ara propietat d'HP. Inicialment era considerat un sistema amb moltes possibilitats, però HP va deixar de desenvolupar dispositius amb aquest sistema i ha alliberat el codi font del sistema.

[25] **Symbian:** Sistema operatiu per a dispositius mòbils, utilitzat sobretot en dispositius Nokia. Era un dels sistemes més utilitzats abans de que iOS i Android s'establissin com les dues grans potències dins de les tecnologies mòbils actuals.

[26] **Bada:** Sistema operatiu mòbil desenvolupat per Samsung, molt minoritari si el comparem amb iOS, Android, o inclús Symbian.

En aquest camp, analitzarem jquery mobile i Sencha touch, ja que estan considerats com dos dels més potents en aquest camp.

Jquery Mobile

Com hem comentat, jquery Mobile és un Framework javascript, optimitzat per a dispositius mòbils i tablets, desenvolupat per l'equip de jquery. L'objectiu d'aquest Framework és tenir un estàndard, de tal manera que el codi estigui adaptat als principals dispositius actuals, sense haver de fer distincions al codi per cada tipus de dispositiu.

Té el gran avantatge que es basa en jquery, el Framework javascript més important actual. La gran majoria de desenvolupadors web el dominen, per tant, la transició a aquesta tecnologia és relativament senzilla.

A més, la combinació amb Phonegap, el transforma en una eina molt potent a priori per al desenvolupament d'Apps multiplataforma. Un codi escrit una vegada es pot utilitzar en els principals sistemes actuals.

La programació és molt senzilla i en molt poques línies de codi es poden obtenir bons resultats. Per a dissenyar l'aspecte visual, s'utilitza HTML5 i CSS3, combinat amb temes que porta ja per defecte jquery mobile i que es poden aplicar simplement assignant les classes corresponents als elements desitjats.

Com a defectes del sistema, podem dir que es tracta d'un sistema que està una fase inicial de desenvolupament. Ha sortit la versió 1.0 del sistema al febrer, amb el que acaba de néixer oficialment. La documentació al respecte és escassa i en moltes ocasions no s'entén del tot bé. Els temes visuals que aporta el Framework són escassos i la programació, tot i ser senzilla, pot arribar a ser caòtica, per la llibertat que ofereix al programador (una mica, el mateix problema que té el jquery convencional).

El problema més greu que té jquery mobile és de rendiment. És significativament més lent que una aplicació nativa, i té problemes importants amb les llistes llargues, on s'ha de permetre a l'usuari fer scroll vertical. El scroll és molt poc fluid, i tot i que és cert que es pot solucionar aquest problema utilitzant un plug-in[27], ja s'han d'utilitzar elements no programats per l'equip desenvolupador del Framework.

Un altre problema important es tracta del seu comportament amb Android. Les diferents versions del sistema (que com sabem té problemes de fragmentació importants, per més informació, mirar l'apartat on s'analitza el SDK d'Android), fan que el programa es comporti de

[27] **Plug-in**: Es tracta d'una aplicació que es relaciona amb una altre per a aportar una funcionalitat nova i molt específica. En aquest cas, es tracta d'un scroll vertical molt més fluid. El plug-in utilitzat es diu iScroll.

maneres realment imprevisibles (a més, hi ha molt poca documentació al respecte). Amb dispositius Iphone, el comportament és més previsible i funciona amb un rendiment acceptable.

Aquest estudi prové de l'experiència que tinc derivada de desenvolupar l'aplicació mòbil de l'altre projecte en el qual he participat dins l'empresa (projecte Groupiest) .

Sencha Touch

En aquest tipus de tecnologies, el principal competidor de jquery mobile és Sencha Touch. Tal i com jquery mobile és una tecnologia derivada de jquery, Sencha Touch és una tecnologia derivada d'ExtJs, un altre Framework web javascript, molt important.

Sencha Touch és un Framework bastant més madur que jquery Mobile. Actualment ja són a la versió 2 del sistema, i es tracta d'un Framework que té una potència visual molt gran. Tant les transicions com els components visuals que estan disponibles per defecte, són molt atractius visualment.

La fluïdesa i rendiment de les aplicacions programades utilitzant Sencha Touch i compilades amb Phonegap és molt major al del jquery Mobile, però tot i així, encara és més lent que les aplicacions natives, i es depèn de les funcionalitats que ofereixi el SDK de Phonegap, que tot i ser molt complet, no porta funcions per a utilitzar tots els components dels dispositius. Tal i com el seu competidor, un gran avantatge que té és la possibilitat de ser multiplataforma. Segons els estudis que s'han consultat per a escriure aquesta secció, Sencha Touch es comporta de manera més estable i fiable en tecnologies diferents que jquery Mobile.

A nivell de codi, l'estructura de Sencha Touch és més organitzada que la de jquery Mobile, ja que en aquest cas, el HTML és mínim i s'especifica tot amb javascript.
El problema més gran és que tant Sencha Touch com el seu "germà" ExtJs, tenen una manera d'estructurar el codi javascript molt diferent a la que estem acostumats els desenvolupadors web i fa que la corba d'aprenentatge d'aquest Framework sigui bastant pronunciada.

Una vegada analitzats ambdós Frameworks podem veure que mentre amb un la programació és molt ràpida i senzilla, el seu rendiment no és massa bo, i que amb l'altre, el rendiment és bo, però l'aprenentatge és complicat.

Ara passarem a analitzar els SDKs dels dos sistemes mòbils més utilitzats a l'actualitat, com són Android i iOS.

Android

Android és actualment un dels dos sistemes operatius mòbils més importants. Està basat en Linux, i el desenvolupa la "Open Handset Alliance", un conglomerat de fabricants, desenvolupadors i operadores liderats per Google. Actualment té un 50.9 % de quota de mercat (quasi el doble del seu gran competidor, el iOS. Estadística del segon trimestre de 2011) i té una comunitat de desenvolupadors molt gran. La seva última versió disponible (a data d'avui), és la 4.0, anomenada també Ice Cream Sandwich.

El codi d'Android va ser alliberat amb llicència Apache, és a dir, el codi font del sistema operatiu, és software lliure. Les aplicacions es distribueixen mitjançant la seva pròpia tenda d'aplicacions, que ara s'anomena Google Play (fins fa poc, era coneguda com a Android Market). En aquesta tenda virtual, hi ha més de 400 000 aplicacions, més de dos terços de les quals són gratuïtes.

La distribució de les aplicacions és molt fàcil, donen molta llibertat als programadors que volen distribuir les seves aplicacions. Tot i que això és molt positiu per al programador, també dóna la possibilitat que es posin a disposició de tothom aplicacions amb malware[28], de les quals s'han detectat uns quants de casos a Google Play.

La programació de les aplicacions, que és la part que més ens interessa, es fa en el llenguatge de programació java, utilitzant el seu SDK. Tot i que la descàrrega del SDK és totalment gratuïta, s'ha de destacar que distribuir les aplicacions a Google Play, no ho és[29].

SDK d'Android

Com he comentat, la programació d'aplicacions per Android es fa amb java, tot i que s'ha d'esmentar que també donen la possibilitat de programar-les en altres llenguatges utilitzant altres eines (com els Frameworks javascript i Phonegap, que s'han explicat abans). Al SDK hi ha totes les llibreries i classes necessàries per explotar tant el sistema operatiu en si, com les funcionalitats de cada terminal en concret.

Compta amb una documentació molt completa, que explica cada component en concret, i també ofereixen una sèrie de tutorials, on es poden veure exemples de codi, amb els quals es poden entendre fàcilment els conceptes bàsics de l'SDK.

Les interfícies gràfiques es poden dissenyar de manera senzilla utilitzant XML, i la programació es fa més o menys senzilla segons el teu coneixement de Java.

Personalment, en aquest punt, considero la programació en Java com un punt molt positiu, ja que es tracta d'un llenguatge molt estructurat i molt net, que si el comparem amb el caos en el qual es pot arribar a convertir javascript, surt, des del meu punt de vista, guanyant.

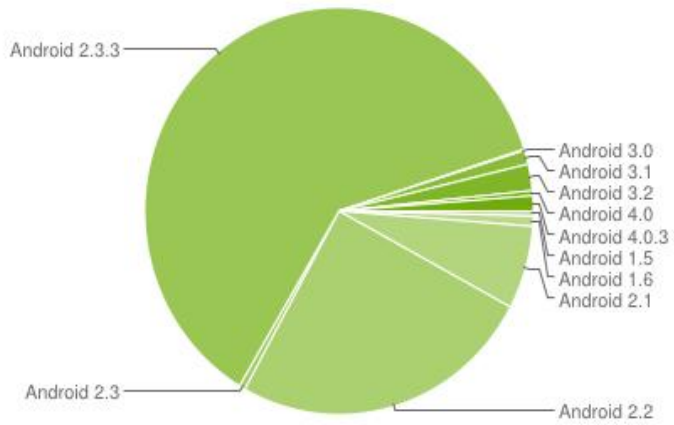
[28] **Malware**: Terme genèric que engloba tot software dissenyat amb propòsits maliciosos.
[29] El cost de poder distribuir les teves aplicacions a Google Play, és de 25 $ (cost només de registre).

L'entorn de programació recomanat, passa per instal·lar-se l'IDE[30] Eclipse[31], amb un plug-in especial que conté suport per al SDK d'Android, i compta amb un emulador[32] incorporat (per a poder provar el codi sense haver de fer-ho en un dispositiu real).

Tot i que Eclipse és un dels IDE més potents de l'actualitat, considero que Android podria tenir un IDE propi per a poder programar, sense haver d'utilitzar tota la base d'Eclipse, que és molt pesada, i pot fer-se lent. Tot i així, es tracta d'un entorn correcte amb el qual programar.

El principal problema amb què s'enfronten a diari els desenvolupadors d'Android, és la fragmentació de versions. Això és un fenomen que s'ha produït en aquest sistema, pel fet que tot fabricant i operadora, modifica el codi font d'Android per a optimitzar-lo per al seu cas en concret. Inicialment pot semblar positiva aquesta personalització, però acaba essent tot el contrari, ja que quan surt una versió nova del sistema, l'actualització no es pot fer fins que l'operadora o el fabricant, adapti la seva versió pròpia d'Android als canvis que s'han produït en la versió més moderna.

Això provoca que els fabricants en alguns casos, deixin els mòbils estancats en una versió antiga del sistema i que no es pugui actualitzar. Amb això, tenim un mercat on el 51% dels dispositius porten Android, però que d'aquest 51% curiosament, molt poca gent té al seu terminal l'última versió del sistema. A continuació es pot veure un gràfic amb la quota de mercat de cada versió d'Android: (estadístiques calculades al març d'aquest any)



[30] **IDE** (*Integrated Development Environment*): És un programa informàtic compost per un conjunt d'eines de programació. Pot ser tant per un llenguatge en concret, com per diversos. Solen comptar amb un editor de codi, un compilador i eines per a debugar codi generalment. Entre els IDEs que podríem destacar hi ha Netbeans, Eclipse, Xcode, o el Visual Studio.

[31] **Eclipse:** IDE de codi obert multiplataforma i multillenguatge. Té un sistema molt potent de plug-ins, que permet instal·lar components específics i personalitzar les funcionalitats del teu IDE depenent de les teves necessitats com a desenvolupador.

[32] **Emulador:** En aquest cas, es tracta d'un programa que forma part del plug-in d'Android de l'Eclipse, que emula el comportament d'un dispositiu mòbil amb Android, és totalment configurable i s'utilitza per a agilitzar la programació i no haver d'estar provant contínuament el codi amb dispositius reals.

Com es pot observar, la versió 2.3.3 és la més utilitzada, pel fet principalment que és l'última versió a la qual molts fabricants permeten actualitzar els seus terminals de gama mitjana[33] (com pot ser per exemple el Samsung Galaxy Ace).

A efectes pràctics, el problema és que no totes les funcionalitats que ofereix el SDK d'Android són compatibles a totes les versions, i si programes amb el SDK de l'última versió, pot passar que alguns terminals que porten una versió anterior, tinguin problemes executant la teva aplicació.

iOS

iOS és el sistema operatiu que porten tots els dispositius mòbils d'Apple, tant els telèfons (Iphone), com els tablets (iPad), com els reproductors mp3 (iPod) d'última generació. La filosofia d'iOS és molt diferent a la d'Android, principalment perquè el codi font és codi tancat, i per tant, no pot esser modificat pels desenvolupadors. Es caracteritza principalment pel fet de tenir molta cura de les interfícies gràfiques i de la fluïdesa de les seves aplicacions, i s'aconsegueix, d'aquesta manera, un rendiment en general molt positiu.

Actualment té aproximadament un 30% del mercat dels smartphones, però s'ha d'observar que la quantitat de dispositius que porten aquest sistema operatiu, en el marc dels smartphones, es limita a totes les versions de l'Iphone, la qual és molt menor a la quantitat immensa de dispositius que porten Android.

Compta també amb una botiga virtual d'aplicacions, anomenada App Store, des de la qual es poden descarregar les més de 500 000 aplicacions que hi ha disponibles. També té una comunitat de desenvolupadors molt gran que distribueix les seves aplicacions a l'App Store (algunes de les quals s'han fet molt populars, com Instagram[34] per exemple). En general, hi ha més aplicacions de pagament que a Android. Els usuaris d'Iphone tenen tendència a pagar per aplicacions més freqüentment que els d'Android.

Per a distribuir aplicacions per a iOS, s'ha de pagar llicència de desenvolupador (que es paga en quotes anuals)[35].

[33] **Terminals de gama mitjana:** En aquest cas, ens referim als smartphones, que tot i no tenir grans especificacions tècniques, compleixen amb les funcionalitats bàsiques dels smartphones i tenen un cost inferior als terminals de gama alta. Tenen una quota molt alta dins el mercat.
[34] **Instagram:** És una aplicació gratuïta per a dispositius Apple, que permet compartir les fotos fetes amb el teu smartphone. Compta amb una sèrie de filtres fotogràfics per a editar les fotografies, i es connecta amb les principals xarxes socials actuals. Actualment, té més de 12 milions d'usuaris i es calcula que s'han compartit més de 100 milions de fotos.
[35] El cost de llicència de desenvolupador d'Apple és de 99 $ per any.

SDK iOS

Les aplicacions per a Iphone, iPad o iPod, es programen generalment en Objective-C, un llenguatge orientat a objectes, i és aquest un superconjunt de C, amb la qual cosa tot el que es considera vàlid en C, ho és també en Objective-C. L'orientació a objectes és similar a la vista en altres llenguatges com Smalltalk[36].

Generalment s'utilitza el Framework Cocoa Touch, que és una capa per damunt Objective-C, que facilita molt la feina a l'hora de programar aplicacions per a iOS i que s'integra perfectament amb l'IDE que proporciona Apple una vegada comprada la llicència de desenvolupador.

En l'aspecte que iOS guanya clarament respecte Android és en les eines proporcionades als desenvolupadors per a programar. Xcode[37], l'IDE que ofereixen és molt potent. Permet gestionar projectes no només per a dispositius mòbils, sinó que també dóna suport per a gran nombre de llenguatges de programació. Permet dissenyar les interfícies gràfiques d'una manera molt senzilla (drag & drop)[38] i en general, podríem dir que és una eina molt còmoda per a treballar.

Personalment, la programació per aquest tipus de dispositius m'atreu, per diverses raons. Inicialment, perquè tot i no haver programat mai en aquest llenguatge, em sembla que la corba d'aprenentatge no pot ser tan alta, gràcies a la bona documentació, i a la similitud a altres llenguatges que ja conec com poden ser Java o C++.

El gran contrapunt que té respecte a Android és que per a treballar, s'ha de fer des d'un sistema Apple per força. Per tant, només podrem programar les aplicacions si utilitzem els ordinadors de la companyia. Bé és cert que teòricament el sistema operatiu es podria virtualitzar[39] utilitzant algun programa com Virtualbox o Vmware, però això no és recomanable, ja que a l'hora de pujar aplicacions a l'App Store, es fan una sèrie de comprovacions per Hardware, que fan que el procés només funcioni si es disposa d'un ordinador de la marca Apple.

[36] **Smalltalk:** Llenguatge de programació on la interacció dels objectes es fa mitjançant l'enviament de missatges.
[37] **Xcode:** IDE desenvolupat per Apple, gratuït, i instal·lat de sèrie amb Mac OS, que permet compilar codi de diversos llenguatges com C, C++, Objective-C o Java. Es tracta d'un IDE molt potent, especialment recomanable per a programar aplicacions mòbils.
[38] **Drag & Drop:** Expressió en informàtica que es refereix a arrossegar elements d'una finestra a una altra o de moure els elements dins una mateixa finestra.
[39] **Virtualitzar:** En aquest cas, ens referim a emular un altre sistema operatiu, executant-se mitjançant un programa com VIrtualbox, que alhora s'executa en el marc d'un sistema operatiu en concret. És a dir, ens referim a executar un sistema operatiu, dins un altre sistema operatiu mitjançant un software de virtualització.

Un altre aspecte negatiu de programar per a iOS és el gran control que porta Apple sobre l'App Store. Tot i que això té el seu costat positiu (no hi ha malware), també pot ser fins i tot molest, ja que si no es segueixen les seves directrius al peu de la lletra, les aplicacions no seran acceptades.

Per a il·lustrar això, amb un exemple es veurà clar. Apple no accepta a l'App Store aplicacions que tinguin un botó que les tanqui. Apple vol que el control de les aplicacions es faci exclusivament des del gestor de tasques del sistema, cosa que fa que tota aplicació que es pugui tancar sola, sigui automàticament rebutjada.

Després d'analitzar una mica les tecnologies que havíem proposat inicialment, podem veure d'una manera més visual un resum de les conclusions que hem extret, a la graella que tenim a continuació:

Sistema	IDE Recomanat	Pros	Contres
Jquery Mobile + Phonegap	Netbeans	Multiplataforma Facilitat de programació Experiència Prèvia	Sistema jove, poc madur Rendiment baix Scroll molt poc fluid Problemes amb Android Programació poc estructurada No te un IDE específic Poca Documentació
Sencha Touch + Phonegap	Netbeans	Multiplataforma Visualment molt potent Codi estructurat Sistema madur, ben documentat	Corba d'aprenentatge molt alta Bon rendiment, però menor a les App natives Cap tipus d'experiència amb aquest sistema No té un IDE específic
SDK Android	Eclipse + plugin Android	Bona documentació Sistema predominant al mercat Experiència en programació Java SDK Gratuit	Fragmentació de versions Públic poc propens a pagar No multiplataforma No té un IDE específic
SDK iOS	Xcode	IDE inclòs amb el SDK i molt potent Bona documentació Públic objectiu molt receptiu Objective-c és similar a Java/C++ Programació estructurada	No Multiplataforma Quota anual per distribuir les App Només es pot programar des de màquines Apple Procés de distribució de les App complex

Amb tota aquesta informació, es pot veure clar que s'optarà per desenvolupar una aplicació nativa, que tot i renunciar a la possibilitat de sortir directament en diverses plataformes, considerem que és la millor opció per a desenvolupar una aplicació amb un bon rendiment i una programació còmoda.

Entre programar per Android o per iOS, només hi ha un factor que pot decantar definitivament la balança. Si a l'empresa no es comptés amb cap ordinador Apple, la decisió seria clarament programar per Android, principalment per la impossibilitat de fer-ho per iOS.

Com que l'empresa disposa d'un dispositiu Apple, la decisió ha estat desenvolupar l'aplicació per a dispositius mòbils Apple.

Hi ha un aspecte del qual no hem parlat encara, i és el perfil dels usuaris dels dos sistemes. Tot i que aquest projecte és només un prototip, i un mitjà per a formar-me, com aquest projecte està emmarcat dins una empresa que, com totes, cerca guanyar diners, s'ha de tenir en compte el perfil dels usuaris dels dos sistemes. S'han consultat diversos estudis i s'ha pogut comprovar que els usuaris de Apple són molt més propensos a pagar per aplicacions que els d'Android. Això és una informació molt interessant, i que fa que ens reafirmem en la nostra decisió de desenvolupar per iOS.

Especificació i Disseny

Visió General

Una vegada finalitzats els estudis previs, l'objectiu d'aquesta fase és especificar i dissenyar tots els components del projecte, tant l'aplicació com la part del servidor per a facilitar el procés posterior d'implementació.

Els continguts que s'exposaran a continuació són els següents:

- Els **Requeriments** del nostre sistema.
- El **Diagrama de casos d'ús del sistema**, amb el desenvolupament de cada un i uns diagrames de seqüència inicials.
- Un **Model Conceptual**, amb les principals classes que interactuaran per tal que l'aplicació funcioni.
- Posteriorment es mostraran els **Diagrames de seqüència**, a nivell de classes, per a veure el funcionament dels casos d'ús d'una manera més detallada.
- El **Diagrama d'arquitectura**, on es veurà esquemàticament tots els components del sistema i les seves interaccions.
- El **Disseny de l'aplicació**, on es veurà d'una manera esquemàtica, com es vol que quedin les pantalles que formaran part de l'aplicació. També es mostrarà un Mapa Navegacional, on es podran veure les transicions entre pantalles.
- L'**especificació de l'API**, que s'encarregarà d'obtenir les dades de la base de dades, i retornar-les en el format correcte a l'aplicació. Es mostraran els contractes de les principals operacions.
- I per últim, una anàlisi dels possibles **Riscos** que es poden materialitzar en el desenvolupament del projecte.

A la següent pàgina, podrem veure els requeriments que el sistema que implementarem haurà de complir. Després ja entrarem en el desenvolupament dels casos d'ús del sistema.

Requeriments del Sistema

Per a concretar el que volem obtenir en aquest projecte, es mostraran a continuació els requeriments que haurà de complir el sistema:

- S'ha de programar una aplicació per a dispositius Iphone que mostri ofertes que l'usuari té al voltant. La posició de l'usuari s'obtindrà mitjançant geolocalització, i es disposarà d'una sèrie d'ofertes geolocalitzades que es mostraran a l'usuari segons la seva posició.

- L'aplicació ha de poder mostrar totes les ofertes en un mapa mitjançant marcadors.

- Les ofertes que s'oferiran seran de la comunitat autònoma de Catalunya exclusivament.

- L'aplicació tindrà una interfície amigable, i serà de fàcil utilització.

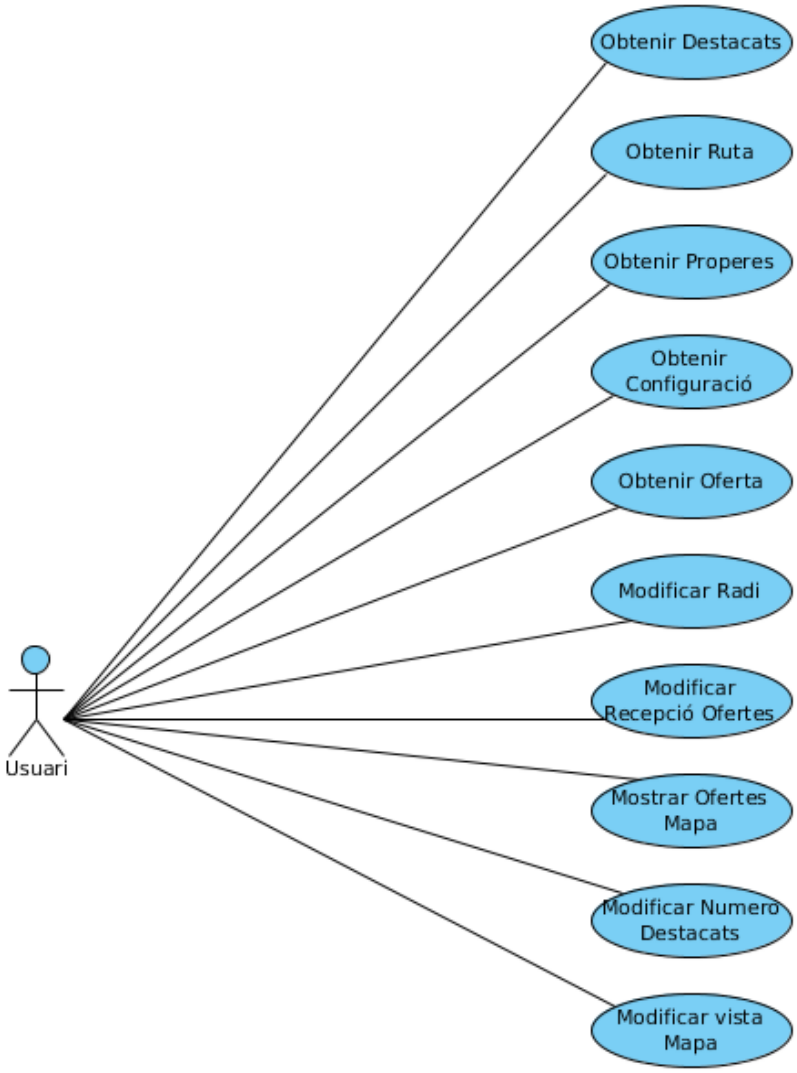
- L'aplicació ha de funcionar de manera fluïda en dispositius Iphone a partir del 3Gs (aquest també està inclòs).

- L'aplicació tindrà un bon rendiment.

- Com a Requeriments opcionals del sistema, hi ha el fet que l'aplicació funcioni amb tablets Apple (Ipad) i que es puguin fer reserves en el cas de les ofertes de Restalo des de la mateixa aplicació.

Diagrama de casos d'ús

A continuació, es mostrarà el diagrama dels casos d'ús del sistema. Podem observar que només tindrem un actor, que en el nostre cas es tractarà d'un usuari que s'hagi descarregat l'aplicació en el seu Iphone. Després de mostrar el diagrama, es detallarà cadascun dels casos d'ús. Primer es mostrarà l'especificació indicant precondicions, postcondicions i els cursos d'esdeveniment de cada un, i posteriorment es veuran els diagrames de seqüència inicials, en els quals veurem una primera aproximació a les interaccions que hi haurà entre els diversos elements del nostre sistema (en aquesta fase, es farà a molt alt nivell, prenent com a elements l'Aplicació, el Servidor, la Base de dades i la Memòria del terminal. Posteriorment ja es concretaran aquests diagrames de seqüència).



Obtenir Destacats

En aquest cas d'ús, s'obtindran les ofertes destacades del sistema. Els criteris que es seguiran per a veure quines són les destacades s'explicaran en detall en l'apartat de la implementació.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari ha seleccionat l'opció "Destacados" del menú inferior de l'aplicació

Postcondicions

- Es mostren tantes ofertes destacades com l'usuari ha seleccionat a les seves preferències, en un llistat.

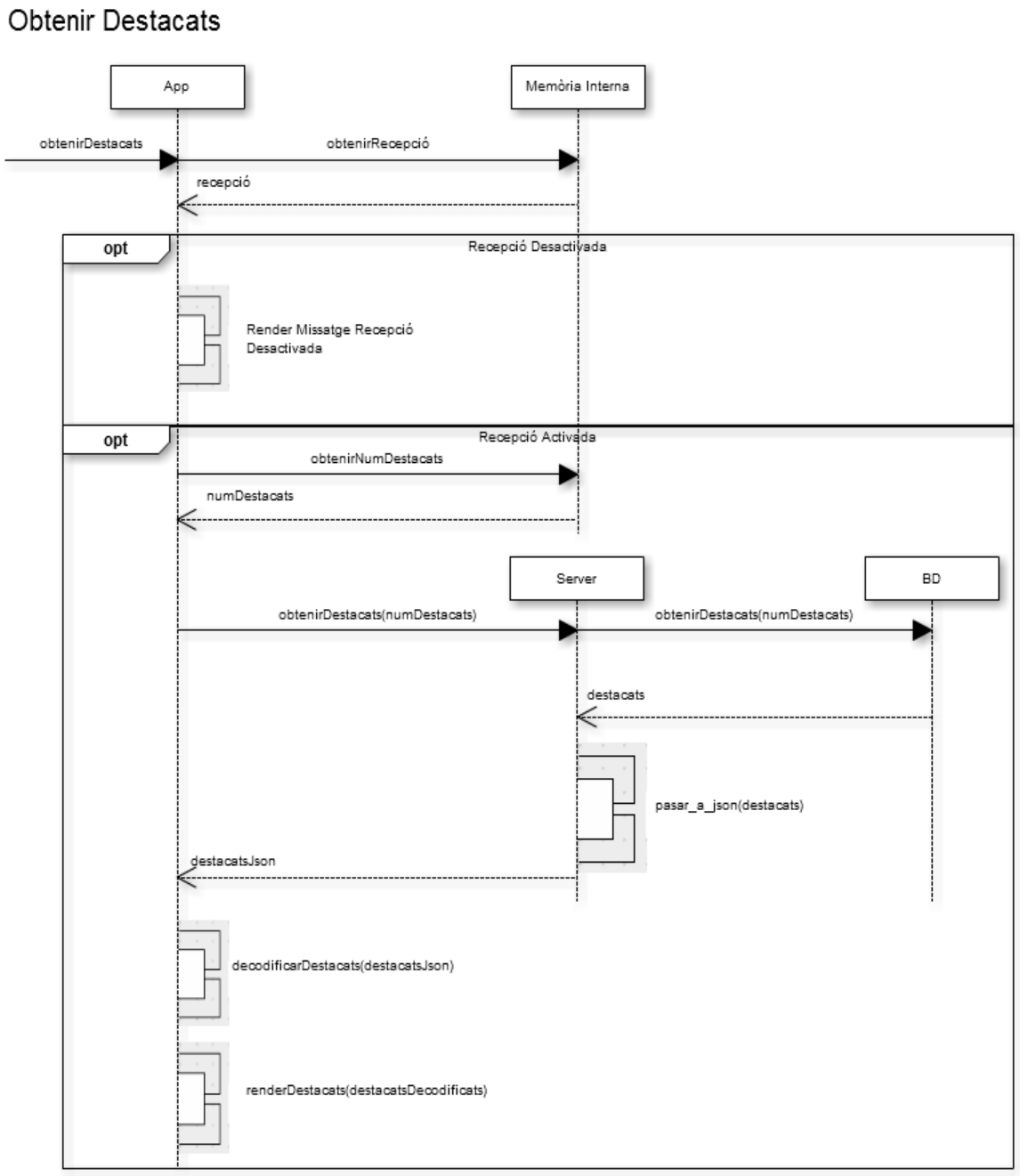
Curs d'esdeveniments bàsic

- L'usuari té la recepció d'ofertes activada.
- L'aplicació obté el número de destacats que l'usuari té seleccionat a la seva configuració.
- L'aplicació fa una crida a l'API, demanant les "n" ofertes destacades (i és "n" el número d'ofertes destacades que vol veure l'usuari).
- L'API accedeix a la Base de Dades, i obté les "n" ofertes destacades. Les agafa i les retorna .
- L'aplicació rep les ofertes destacades, tracta les dades i les mostra en forma de llistat a l'usuari.

Curs Alternatiu

- L'usuari no té la recepció d'ofertes activada.
- L'aplicació mostra un llistat buit, i un missatge informant que té la recepció desactivada.

Diagrama de Seqüència



Com s'ha vist al diagrama, l'aplicació consulta les preferències de l'usuari per saber si ha d'obtenir els destacats o no inicialment. Si la recepció està desactivada, simplement mostrarà un missatge informatiu. Si no és així, mirarà quants resultats demanar, els demanarà, i se li seran retornats en JSON. Una vegada descodificats, simplement haurà de mostrar el llistat per pantalla.

Obtenir Ruta

En aquest cas d'ús, el que es farà serà mostrar la posició de l'usuari i de l'oferta seleccionada en un mapa, i mostrar una ruta entre la posició de l'usuari i l'oferta.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari té la recepció d'ofertes activada.
- L'usuari ha seleccionat una oferta de la llista de destacats, de la de les ofertes que tens a prop teu, o ha clicat en un marcador del mapa de la tercera pestanya.
- L'usuari ha clicat al botó "Cómo llegar" de la vista de l'oferta seleccionada.

Postcondicions

- Es mostra en un mapa, que té el tipus de visualització que l'usuari ha indicat a la configuració, la posició de l'usuari, la de l'oferta amb un marcador i una ruta dibuixada, entre l'usuari i l'oferta.

Curs d'esdeveniments bàsic

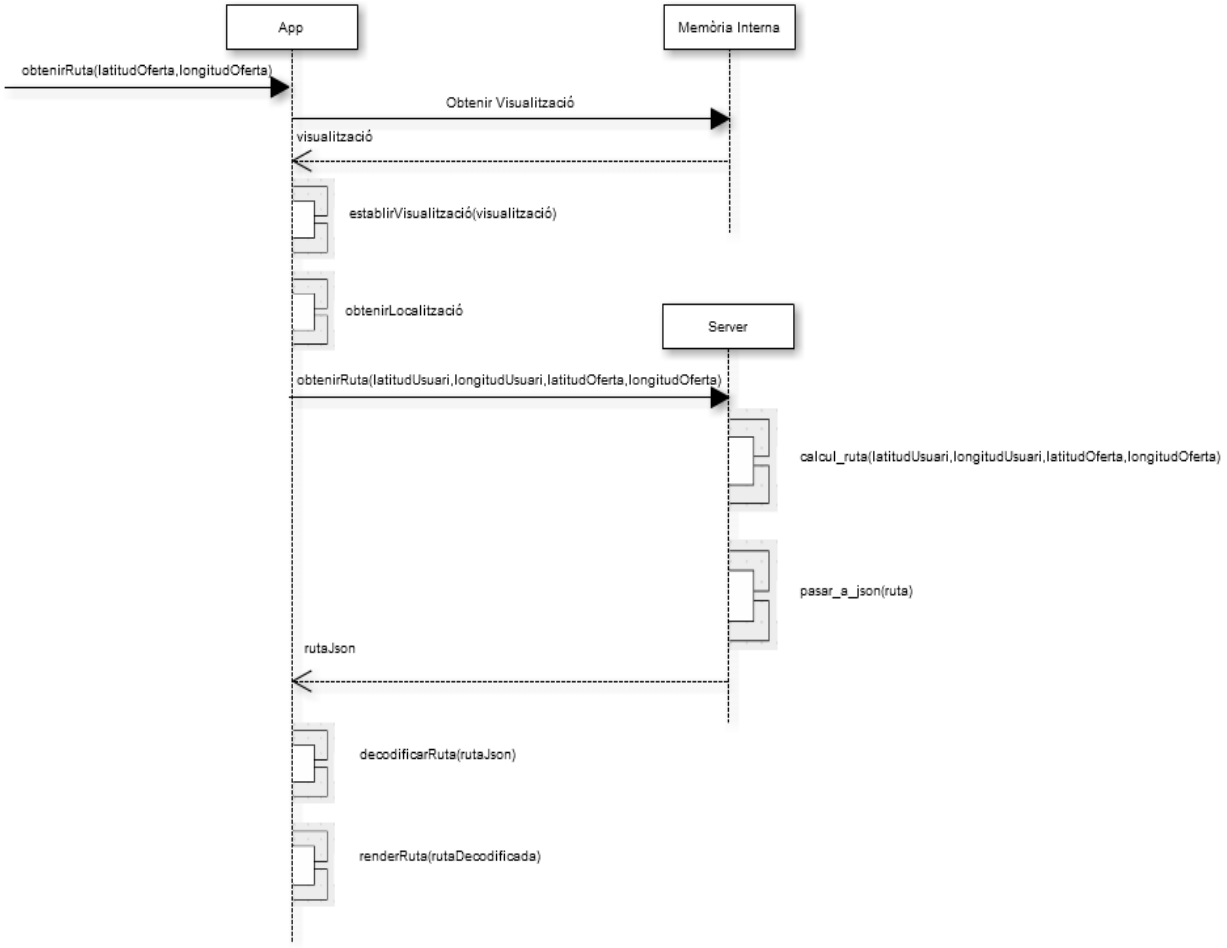
- L'aplicació obté la visualització seleccionada per l'usuari a la configuració.
- L'aplicació mostra un mapa amb la visualització seleccionada.
- L'aplicació mostra la posició de l'usuari al mapa.
- L'aplicació representa la posició de l'oferta amb un marcador al mapa.
- L'aplicació calcula la ruta entre l'usuari i l'oferta.
- L'aplicació dibuixa la ruta calculada.

Curs Alternatiu

- No aplica

Diagrama de Seqüència

Obtenir Ruta



En aquest cas, el diagrama de seqüència és bastant senzill. L'aplicació obtindrà la localització de l'usuari i l'enviarà, juntament amb la localització de l'oferta (que ens arribarà per paràmetre, pel fet que aquesta vista només és accessible passant per la de l'oferta) al Servidor, que calcularà la ruta i la retornarà en format Json. Descodificarem aquesta informació i només haurem de mostrar-ho per pantalla.

Obtenir Properes

Aquí es mostrarà a l'usuari les ofertes que té al seu voltant en forma de llistat.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari ha clicat al botó de la barra inferior anomenat "Cerca de Ti".

Postcondicions

- L'aplicació mostra a l'usuari un llistat ordenat de manera creixent per distància de les ofertes que són dins el radi de recepció definit a la configuració de l'aplicació.

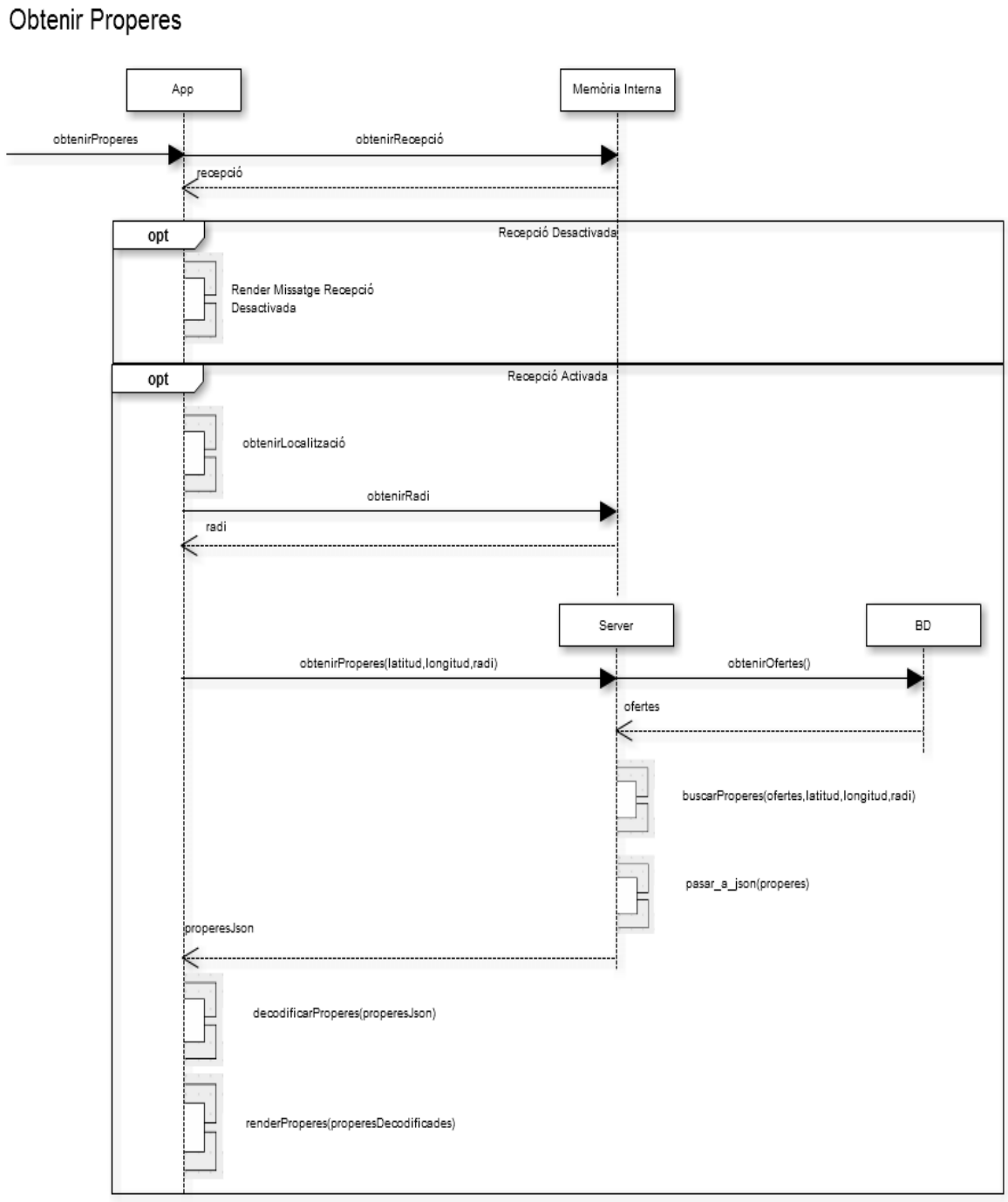
Curs d'esdeveniments bàsic

- L'usuari té la recepció d'ofertes activada.
- L'aplicació obté la posició de l'usuari.
- L'aplicació obté el radi definit per l'usuari.
- L'aplicació crida a l'API amb la posició de l'usuari i el radi com a paràmetres.
- L'API busca a la base de dades les ofertes dins del radi i les retorna.
- L'aplicació agafa les dades, les tracta i les mostra en un llistat ordenat de manera creixent per distància a l'usuari.

Curs Alternatiu

- L'usuari no té la recepció d'ofertes activada.
- L'aplicació mostra un llistat buit i un missatge informant que té la recepció desactivada.

Diagrama de Seqüència



Aquest segurament sigui el diagrama de seqüència més complex d'aquesta fase. Podem veure que comença com el d'obtenir els destacats, obtenint si ha de fer o no la crida (segons la recepció si està activada o desactivada). Si està activada, obtindrem el radi i la posició de l'usuari i els passarem al servidor, que buscarà les ofertes que hi ha dins el nostre radi i les retornarà. Les dades retornades es descodificaran i mostraran per pantalla en forma de llistat ordenat de manera creixent per distància.

Obtenir Configuració

Aquí es mostrarà a l'usuari la configuració que té establerta.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari ha clicat al botó de la barra inferior anomenat "Config".

Postcondicions

- L'aplicació mostra a l'usuari els paràmetres que té definits com a les seves preferències.

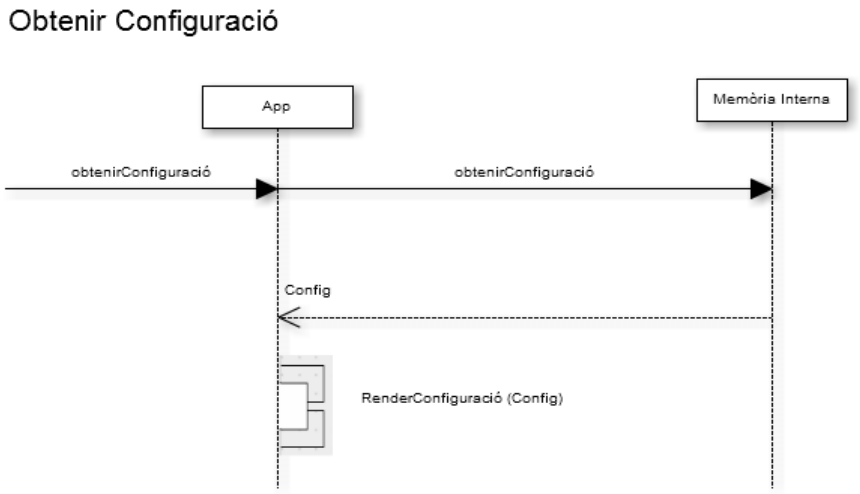
Curs d'esdeveniments bàsic

- L'aplicació busca a la memòria del terminal les preferències de l'usuari i les troba.
- L'aplicació mostra les preferències de l'usuari, és a dir, si la recepció d'ofertes està activada o no, quina vista té com a establerta per al mapa, quantes ofertes destacades se li mostraran i quin és el radi de recepció que té establert.

Curs Alternatiu

- L'aplicació busca en la memòria del terminal les preferències de l'usuari, però encara no han estat definides.
- L'aplicació mostra les opcions per defecte del sistema.

<u>Diagrama de Seqüència</u>



Aquest diagrama és bastant explicatiu per ell mateix. S'obté la configuració de memòria interna i es mostra, simplement.

Obtenir Oferta

En aquest cas, mostrarem la informació d'una oferta en concret.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari té la recepció d'ofertes activada.
- L'usuari ha clicat o bé en una oferta del llistat de les destacades o en el llistat de les que té a prop seu o bé en un marcador del mapa.

Postcondicions

- L'aplicació mostra a l'usuari el contingut complet d'una de les ofertes del sistema.

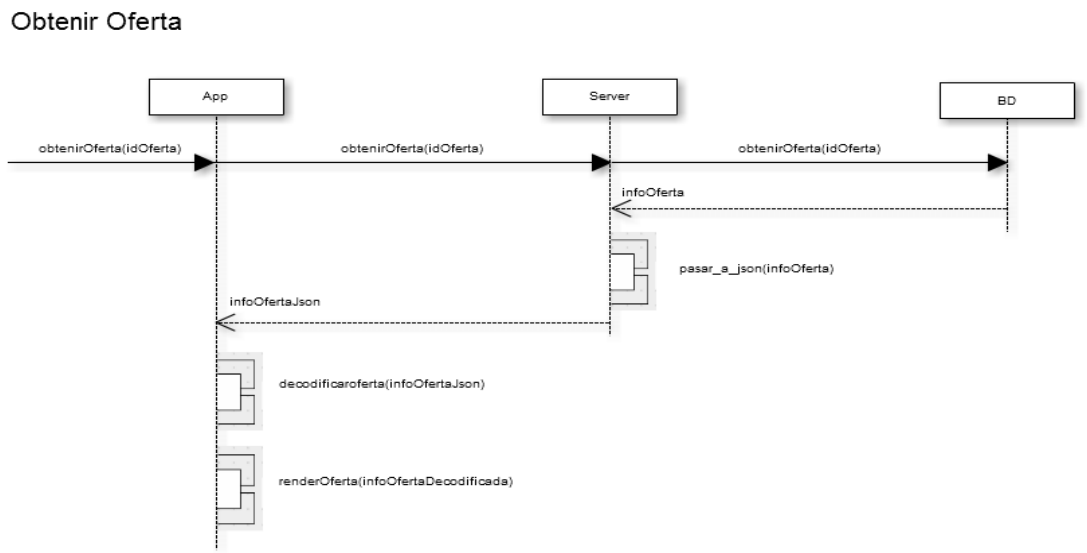
Curs d'esdeveniments bàsic

- L'aplicació demana al servidor les dades completes de l'oferta en concret.
- El servidor agafa les dades d'aquella oferta en concret i les retorna.
- L'aplicació tracta les dades retornades i les mostra per pantalla.

Curs Alternatiu

- No Aplica

<u>Diagrama de Seqüència</u>



En aquest cas, l'objectiu és obtenir la informació d'una oferta en concret. S'obtindrà aquesta informació demanant-la al servidor. Es descodificarà i es mostrarà la informació a la vista de l'oferta.

Modificar Radi

En aquest cas d'ús, l'usuari modificarà el radi d'acció que vol tenir per a la recepció de les ofertes.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari ha clicat al botó de la barra inferior anomenat "Config".
- L'usuari ha modificat el valor del Radi de recepció de les ofertes del sistema.

Postcondicions

- L'aplicació modifica el radi d'acció, o sensibilitat, del sistema, cosa que serà guardada amb les preferències de l'aplicació a la memòria interna del dispositiu.

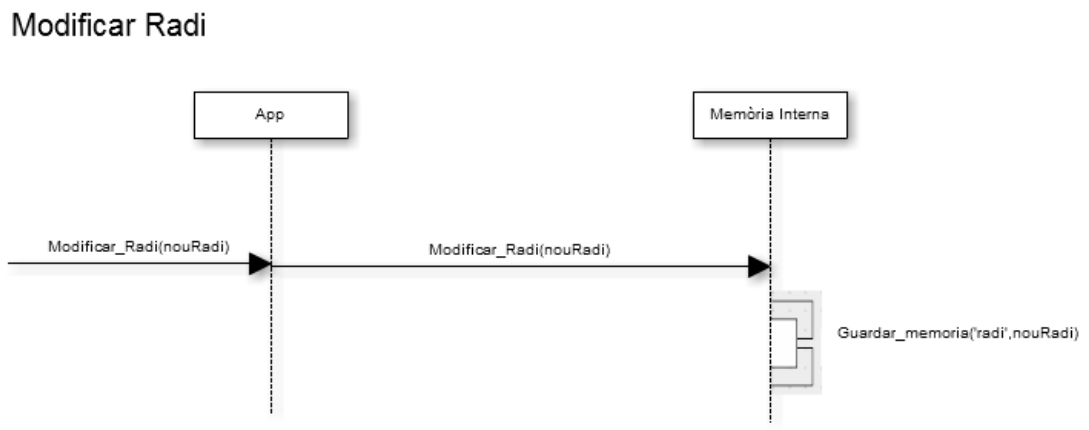
Curs d'esdeveniments bàsic

- L'aplicació obté el valor nou del radi.
- L'aplicació modifica aquest valor, en les preferències del sistema de l'usuari.

Curs Alternatiu

- No aplica

<u>Diagrama de Seqüència</u>



Modificar Recepció d'Ofertes

En aquest cas, l'usuari tindrà la capacitat de canviar l'estat de la recepció d'ofertes. És a dir, podrà activar la recepció si està apagada, o apagar-la si està encesa.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari ha clicat al botó de la barra inferior anomenat "Config".
- L'usuari ha modificat el valor de la recepció d'ofertes.

Postcondicions

- L'aplicació ha modificat la recepció de les ofertes.

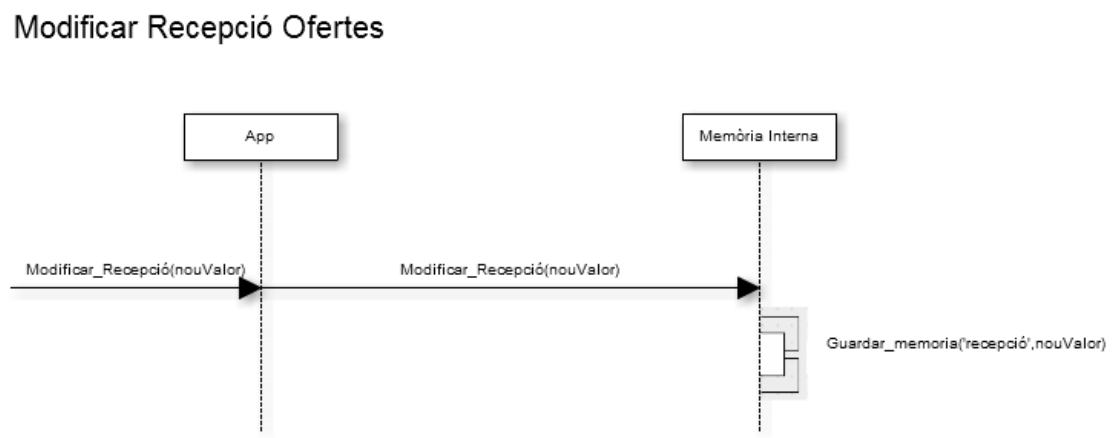
Curs d'esdeveniments bàsic

- L'usuari té la recepció activada.
- L'aplicació desactiva la recepció d'ofertes.

Curs Alternatiu

- L'usuari té la recepció desactivada.
- L'aplicació activa la recepció d'ofertes.

<u>Diagrama de Seqüència</u>



Mostrar Ofertes Mapa

En aquest cas, l'usuari veurà en un mapa totes les ofertes que hi ha al sistema.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari ha clicat al botó de la barra inferior anomenat "Mapa".

Postcondicions

- L'aplicació ha mostrat, utilitzant Markers, totes les ofertes del sistema en un mapa que té com a tipus de visualització, la que l'usuari ha seleccionat a les seves preferències.

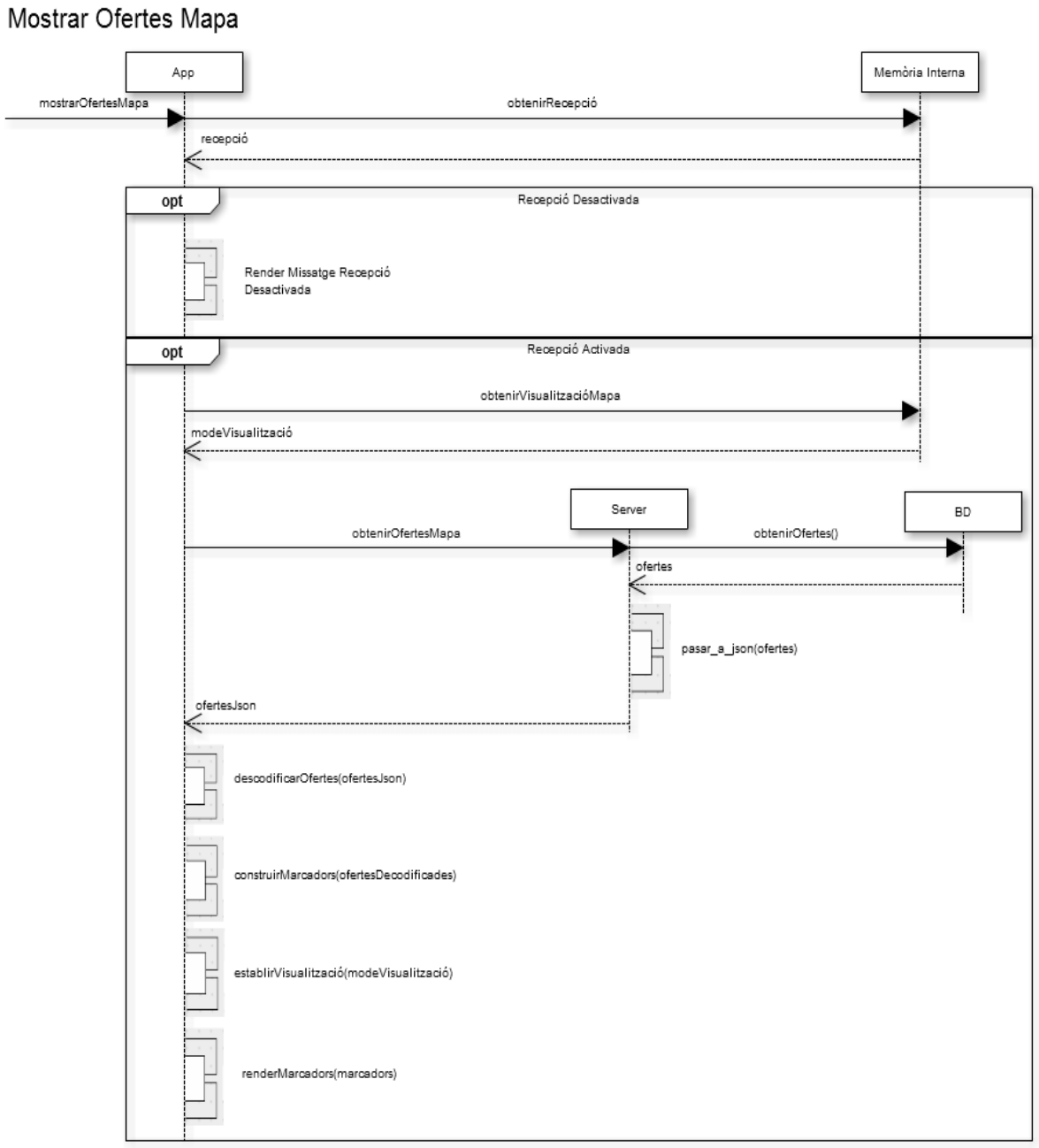
Curs d'esdeveniments bàsic

- L'usuari té la recepció activada.
- L'aplicació obté la visualització seleccionada per l'usuari a la configuració..
- L'aplicació mostra un mapa.
- L'aplicació mostra la posició de l'usuari.
- L'aplicació crida a l'API i demana les ofertes del sistema.
- L'API retornarà les ofertes corresponents.
- L'aplicació mostrarà les ofertes al mapa utilitzant Markers.

Curs Alternatiu

- L'usuari té la recepció desactivada.
- L'aplicació no mostra el mapa i mostra un missatge que informa del fet que té la recepció desactivada.

Diagrama de Seqüència



En aquest punt, el nostre objectiu és representar la informació que tenim a la base de dades sobre un mapa. Per a fer això (sempre que la recepció estigui activada), farem una petició al servidor que ens retornarà les ofertes del sistema, les descodificarem, i a partir d'aquesta informació generarem un marcador per oferta. Establirem la visualització que l'usuari tingui a les seves preferències i mostrarem els marcadors sobre el mapa. A més, es mostrarà la posició de l'usuari al mapa també (en aquest cas, considerem que es farà a renderMarcadors, però evidentment la posició de l'usuari es veurà d'una altra manera, per a distingir-lo de les ofertes).

Modificar Número de Destacats

En aquest cas, l'usuari modificarà el número d'ofertes que se li mostraran a l'apartat de les ofertes destacades.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari ha clicat al botó de la barra inferior anomenat "Config".
- L'usuari ha modificat el valor del número de destacats.

Postcondicions

- L'aplicació ha modificat el número d'ofertes destacades que es mostraran i ho ha emmagatzemat a la configuració de l'usuari a la memòria interna del terminal.

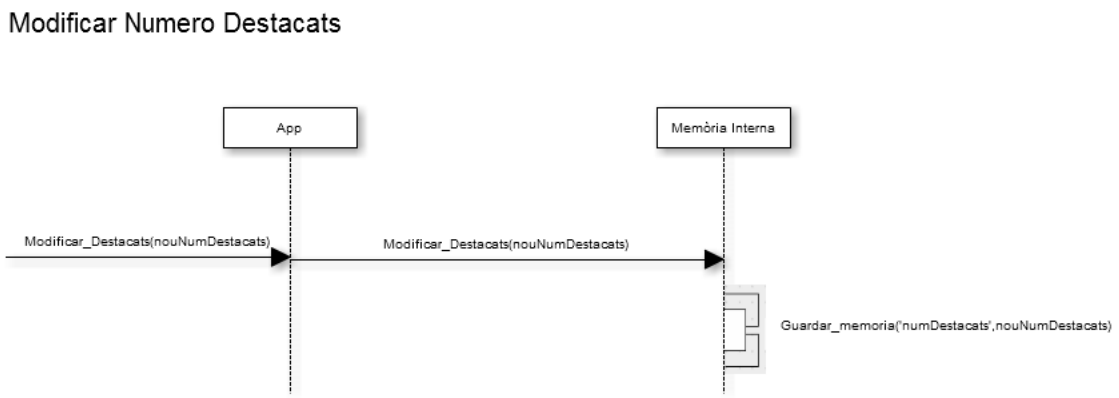
Curs d'esdeveniments bàsic

- L'aplicació obté el nou número de destacats.
- L'aplicació ho emmagatzema amb les preferències de l'usuari a la memòria interna del terminal.

Curs Alternatiu

- No Aplica.

<u>Diagrama de Seqüència</u>



Modificar Vista Mapa

En aquest cas, l'usuari podrà canviar el tipus de visualització amb el que es mostraran els mapes de l'aplicació.

Precondicions

- L'usuari té l'aplicació encesa.
- L'usuari ha clicat al botó de la barra inferior anomenat "Config".
- L'usuari ha modificat el tipus de mapa.

Postcondicions

- L'aplicació ha modificat la visualització dels mapes de les preferències de l'usuari.

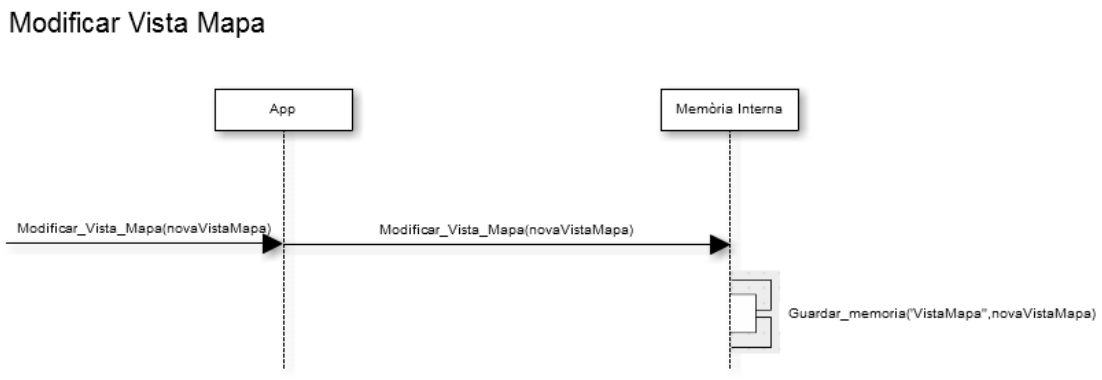
Curs d'esdeveniments bàsic

- L'aplicació obté la nova visualització seleccionada.
- L'aplicació ho guarda a la memòria interna del dispositiu.

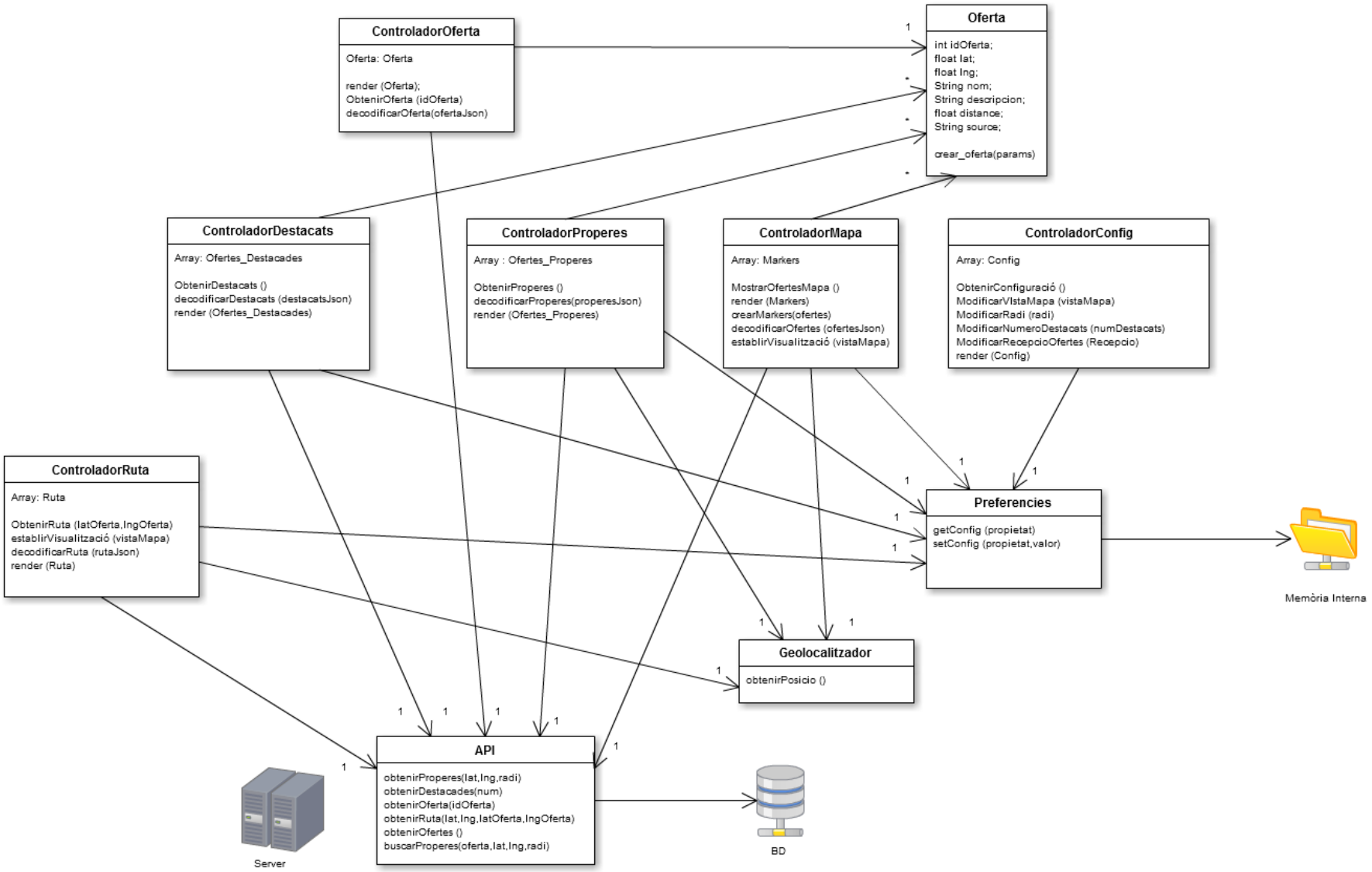
Curs Alternatiu

- No Aplica

<u>Diagrama de Seqüència</u>



Model Conceptual



Abans hem vist els casos d'ús del sistema, amb la seva especificació i uns diagrames de seqüència que ens han servit per a veure com els casos d'ús feien que els principals components del sistema interactuessin. A partir d'això, s'ha elaborat un diagrama conceptual, que ja s'ha mostrat. Com a comentaris al respecte cal deixar clar que les classes que anomenem "controlador", són les que gestionen les vistes del nostre sistema. Així, el ControladorRuta és qui gestiona la vista de les rutes de la qual hem parlat en l'apartat del disseny de la interfície. Aquestes vistes no s'han mostrat per guanyar en claredat, i simplement, al diagrama tindrien una relació amb el seu controlador.

S'han indicat també les navegabilitats, i com veiem, hi ha dues classes molt utilitzades. Aquestes classes són Preferències i Oferta. Preferències serà la classe que actuarà com a intermediari amb la memòria interna del terminal. D'aquesta manera, sempre que es vulgui consultar un element emmagatzemat a la memòria del terminal, s'haurà d'utilitzar una instància d'aquesta classe. Per a fer-ho, disposem d'un mètode anomenat "getConfig", que admet un paràmetre. Aquest paràmetre, si no és nul, serà un String que indicarà quina propietat volem. Si és nul, retornarem totes les preferències.

Quant a la classe Oferta, aquesta és una classe que s'utilitza per a encapsular el que ens retorna l'API en objectes, per a poder manejar d'una manera més senzilla la informació. Veiem que algunes classes utilitzen entre zero i moltes instàncies d'aquesta classe. Això és perquè quan l'API ens retorna més d'una oferta, el que es farà serà crear un Array d'objectes Oferta, que s'anirà recorrent per a mostrar la informació a l'usuari d'una manera senzilla.

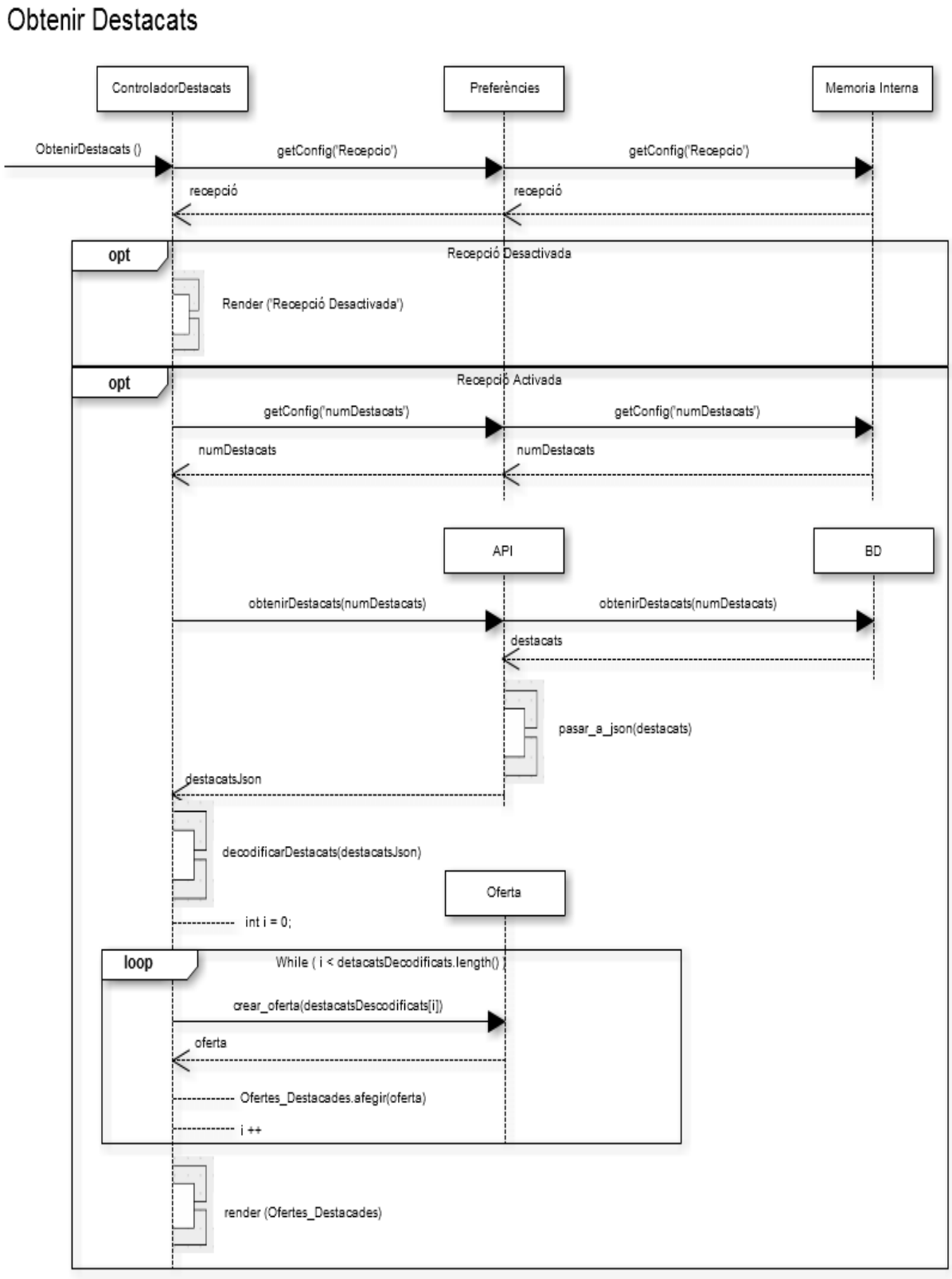
Hi ha també una classe anomenada Geolocalitzador, que tindrà com a funció gestionar l'obtenció de la posició de l'usuari.

Per altra banda, veiem una classe API, que serà a la que els controladors accediran mitjançant crides al servidor. Aquesta classe serà l'única que accedirà a la base de dades i recordem que estarà implementada en php.

Ara mateix, que ja hem desglossat la part dels diagrames anteriors, que anomenàvem "App" en classes, podem refer els diagrames, per a veure en un nivell de detall molt més alt com funcionaran els casos d'ús.

A continuació farem exactament això, mostrar els diagrames de seqüència dels casos d'ús amb l'estructura de classes que acabem de veure.

Diagrames de Seqüència



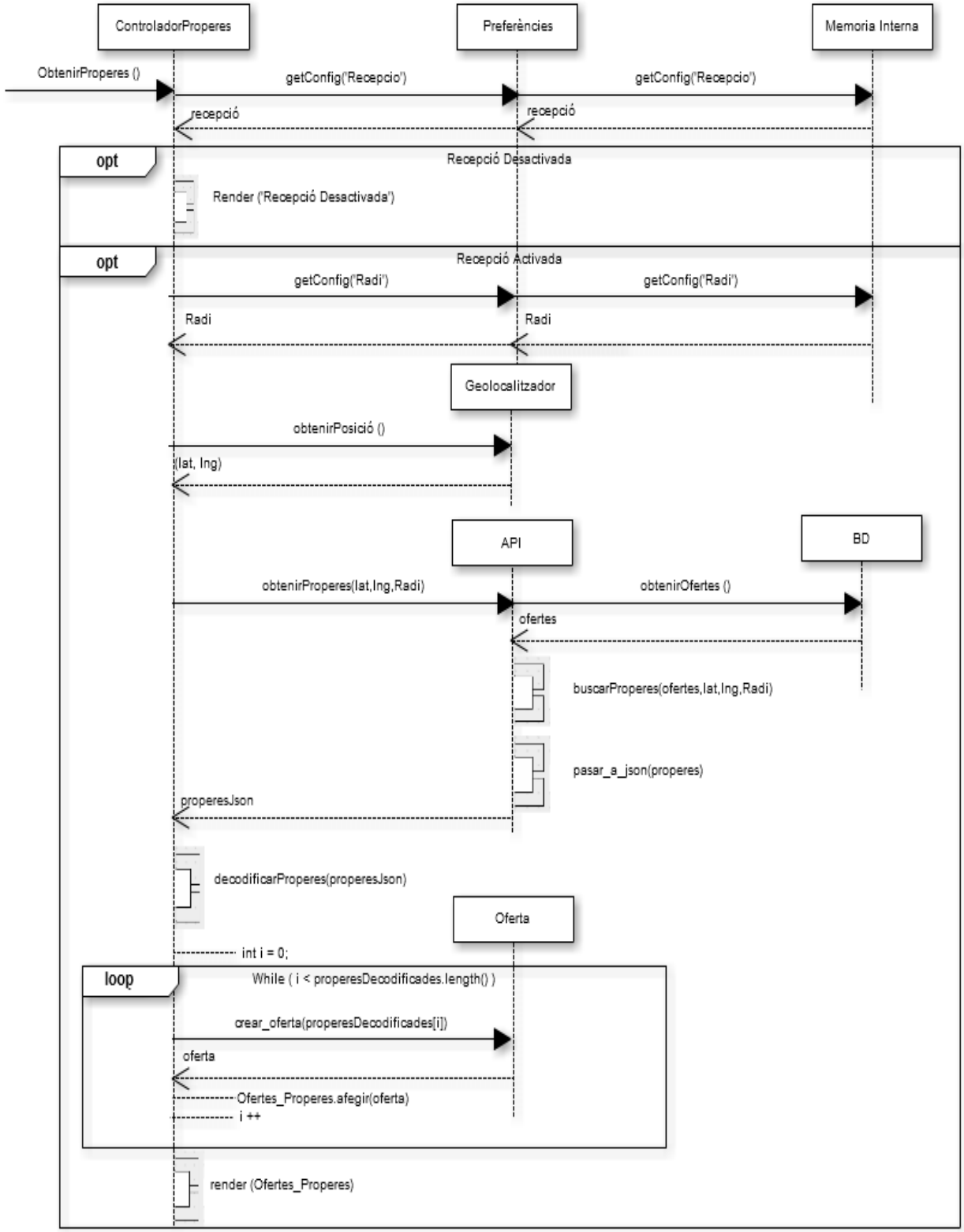
Recordem que el que es fa en aquest cas, és buscar els destacats del nostre sistema. Per a fer això, haurem de veure quants dels destacats vol veure l'usuari i demanar aquesta quantitat de resultats a l'API. Posteriorment, s'haurà de tractar el retorn i mostrar a la interfície.

En el diagrama anterior, ja podem veure com les classes interactuen. Podem observar que la classe Preferències és la que s'instancia i s'utilitza per a accedir a les preferències de l'usuari emmagatzemades en la memòria del dispositiu.

Un altre detall interessant és com es gestionen els resultats que ens retorna l'API. Es pot observar com les ofertes, una vegada descodificades, es van encapsulant utilitzant la classe oferta, i afegint a un Array, que com hem vist en el model, és un atribut de la classe. Una vegada construït l'Array d'objectes corresponent a les ofertes destacades del nostre sistema, simplement s'ha de cridar a la funció render, que, com veurem, és una funció que tenen tots els Controladors que s'encarrega de fer canvis en l'aspecte visual.

En aquest cas, el que farà la funció render serà agafar les ofertes destacades i mostrar-les a l'usuari en forma de llista, tal i com s'ha mostrat en l'apartat de disseny de les vistes.

Obtenir Properes

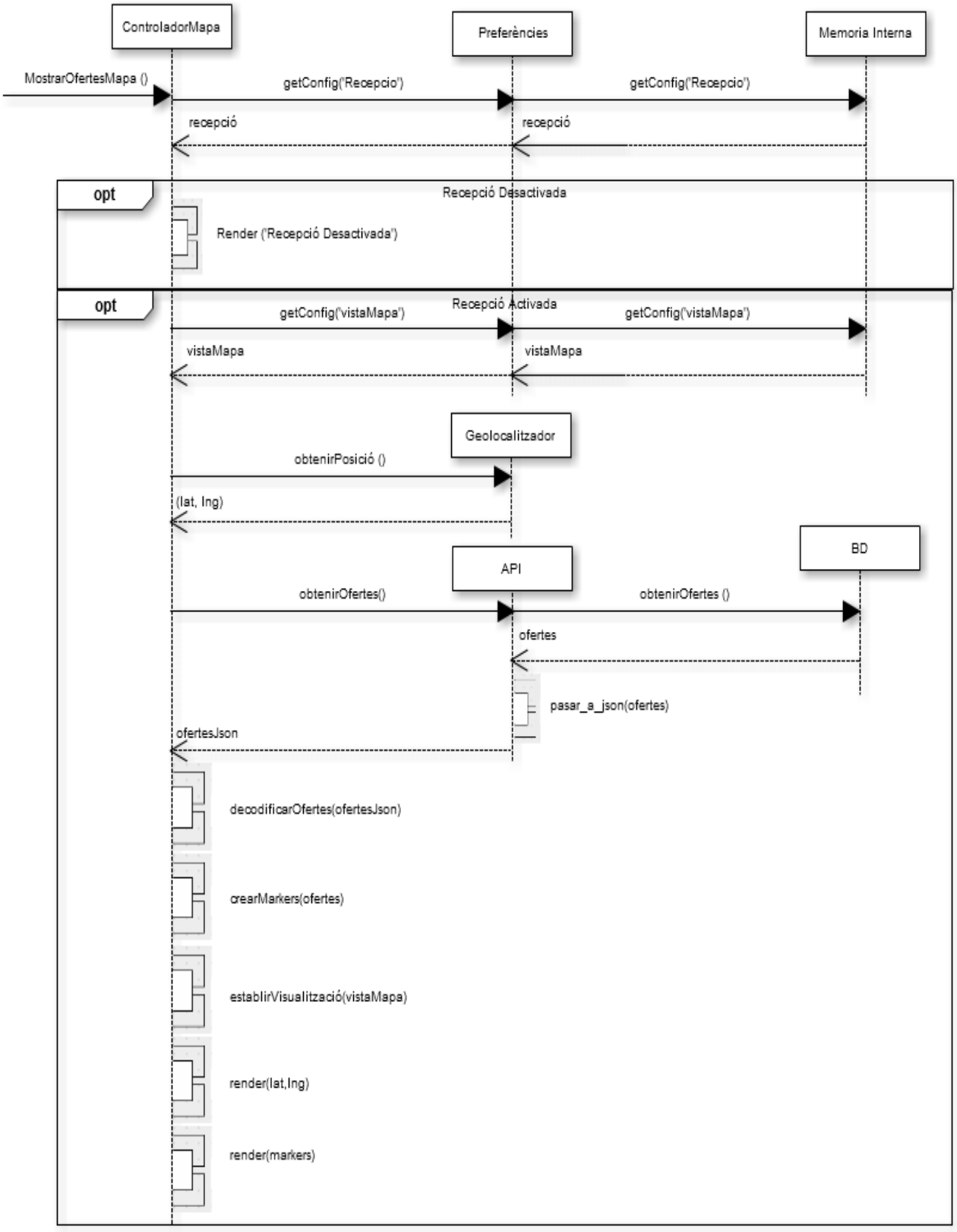


L'objectiu d'aquest cas d'ús és obtenir les ofertes que són a prop de l'usuari. La locució "a prop" la mesurarem en funció del radi d'acció que l'usuari hagi establert en les seves preferències. D'aquesta manera, només es mostraran els elements que estiguin a una distància menor o igual al radi especificat.

El flux del diagrama és molt similar al mostrat anteriorment, si la recepció d'ofertes està activa s'agafa el radi i la posició i s'envien a l'API, que agafarà de totes les ofertes que hi ha al sistema, les que són a una distància menor o igual al radi. Aquestes dades s'ordenaran de manera ascendent en funció de la distància i es retornaran en Json, es descodificaran i encapsularan en un Array d'objectes de tipus Oferta (tal i com ho hem fet al cas d'ús anterior).

Finalment només ens quedarà fer el render corresponent a aquest llistat, per a mostrar les ofertes en forma de llistat ordenat de menor a major distància a l'usuari (sempre dins els límits del radi especificat).

Mostrar Ofertes Mapa



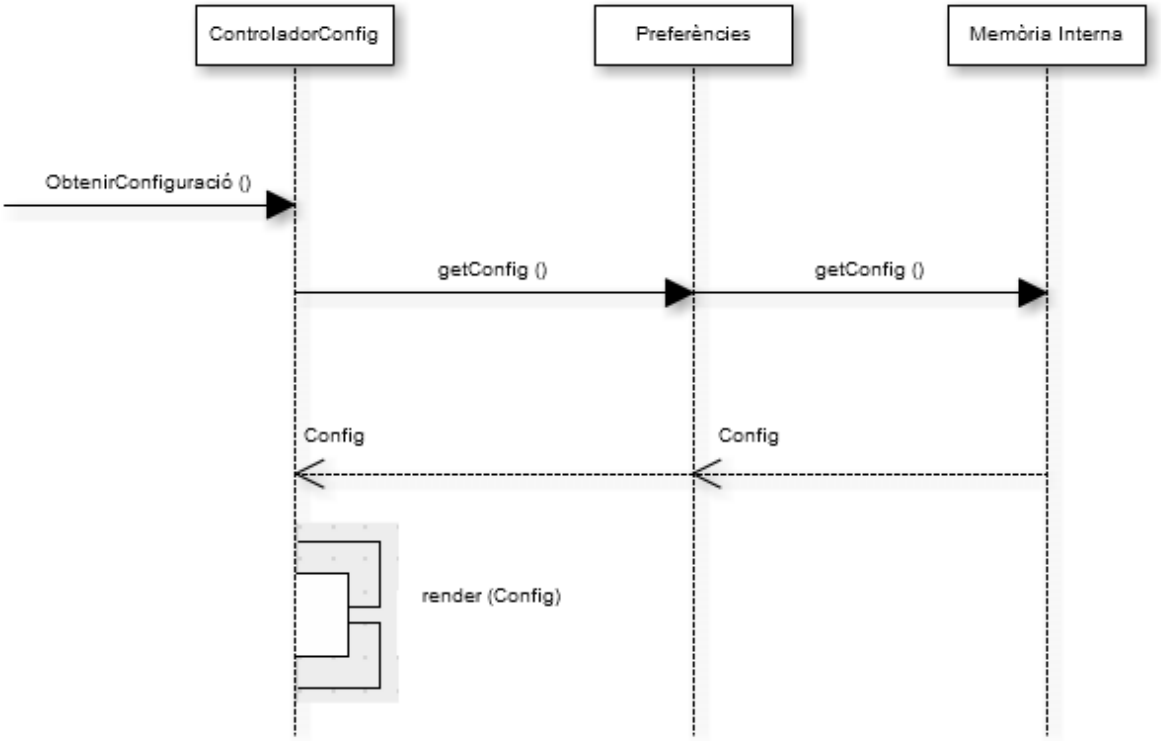
Aquí l'objectiu és mostrar totes les ofertes del sistema en un mapa. Per això, haurem de fer la crida corresponent a l'API, sempre que la recepció d'ofertes estigui activa. Una vegada s'hagin obtingut les ofertes, s'haurà de generar un Array amb els Markers, que seran els objectes que es representaran sobre el mapa.

S'haurà d'obtenir la visualització que té establerta l'usuari (utilitzant la classe Preferències per a fer-ho), i fer que el mapa es mostri d'aquesta manera.

A més, haurem de mostrar quina és la posició de l'usuari. Això és el que es vol expressar quan es fa "Render(lat,lng)", això simplement representa afegir un Marker amb la posició de l'usuari.

Finalment, s'haurà de fer el render dels Markers. El que obtindrem és un mapa amb el mode desitjat per l'usuari, en què es podrà veure la seva posició i la de les ofertes del sistema.

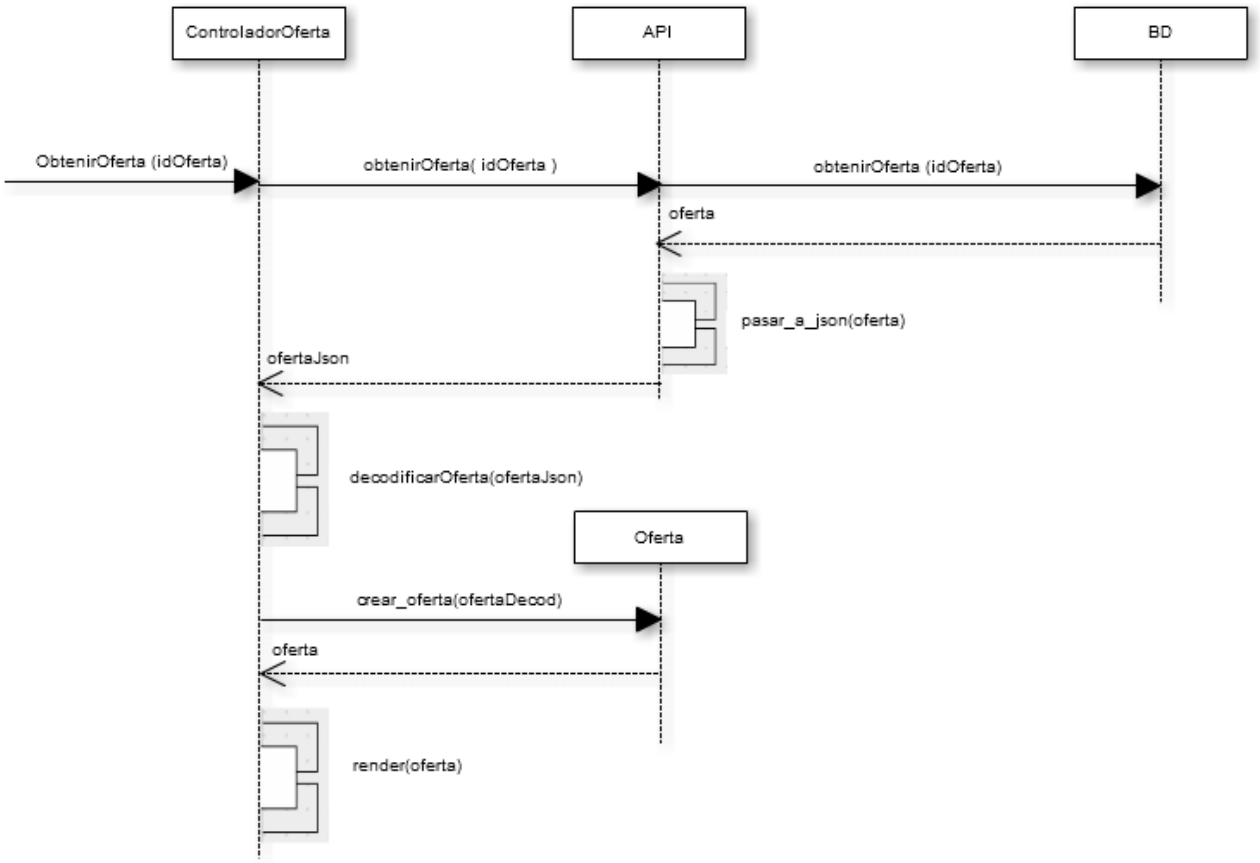
Obtenir Configuració



El que volem aconseguir aquí és obtenir la configuració de l'usuari i mostrar-la per pantalla. Això, com podem veure, es pot fer d'una manera molt senzilla, instanciant la classe Preferències, i utilitzant el mètode getConfig, sense passar-li cap paràmetre, mode en què retorna tota la configuració. Aquesta configuració, finalment, es mostrarà a nivell d'interfície a l'usuari.

S'ha de comentar que si és la primera vegada que s'utilitza l'aplicació, no trobarà configuració. En aquests casos, el que es farà serà mostrar una configuració per defecte que definirem.

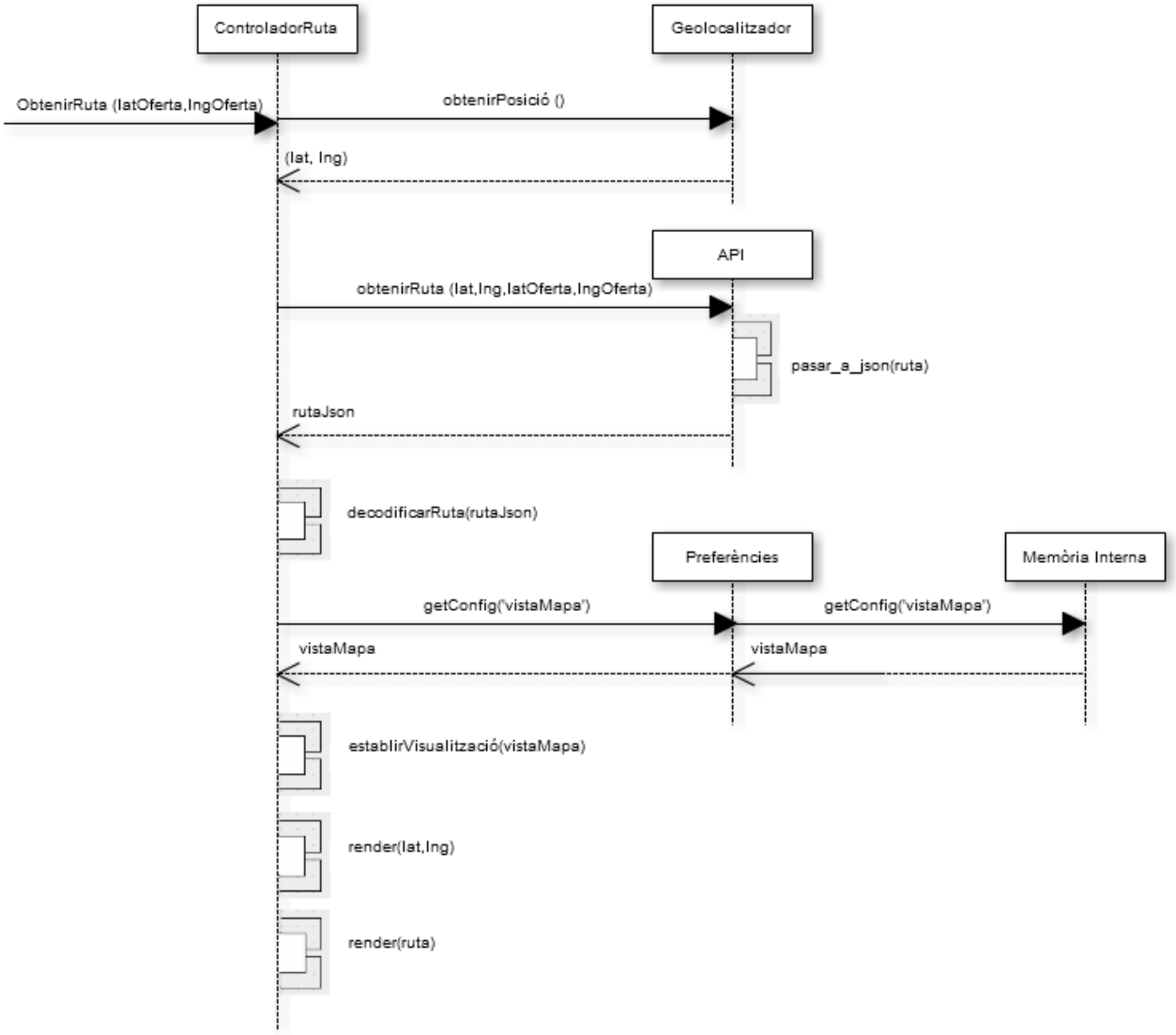
Obtenir Oferta



El que volem obtenir en aquest punt és una oferta en concret, per a mostrar-la a la seva vista pròpia. Per a fer-ho, podem veure que en rebrem la id, això es pot justificar d'una manera senzilla. A aquesta vista, accedirem necessàriament des d'una altra. No es pot accedir a aquesta vista sense passar per la dels Destacats, la de les ofertes properes o la del mapa. Per tant, el paràmetre el rebrà d'aquestes vistes.

El procés podem veure que és molt simple. Simplement busquem l'oferta mitjançant l'API, i la retornem. Com sempre, això és descodificat i l'objecte corresponent de tipus Oferta es crea, per a finalment mostrar-lo a l'usuari, mitjançant la funció render.

Obtenir Ruta



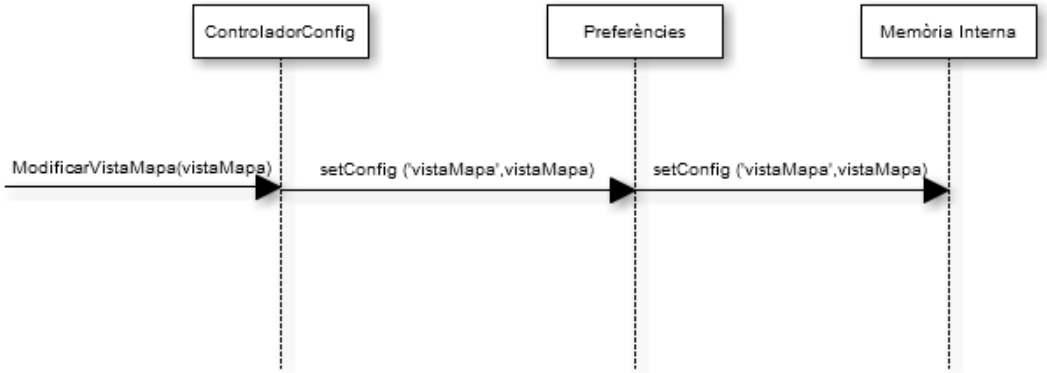
Aquest és el cas en el qual volem calcular la ruta entre la posició de l'usuari i la d'una oferta en concret. Com que només podem accedir a la vista associada passant per la vista d'una oferta en concret, sabrem ja la seva posició.

A partir d'això, el que farem serà obtenir la posició de l'usuari i demanar-li que ens calculi la ruta, cosa que detallarem en l'apartat de la implementació del nostre sistema.

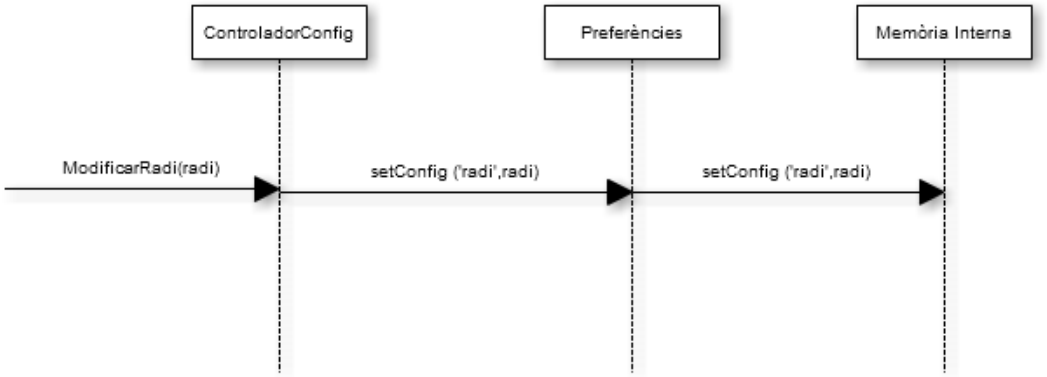
Posteriorment, establirem la visualització que l'usuari hagi elegit en les seves preferències, mostrarem la posició de l'usuari, i la ruta, que alhora mostrarà, com a punt final, l'oferta.

Els diagrames restants considero que són molt explicatius per ells mateixos, per això no comptaran amb justificació.

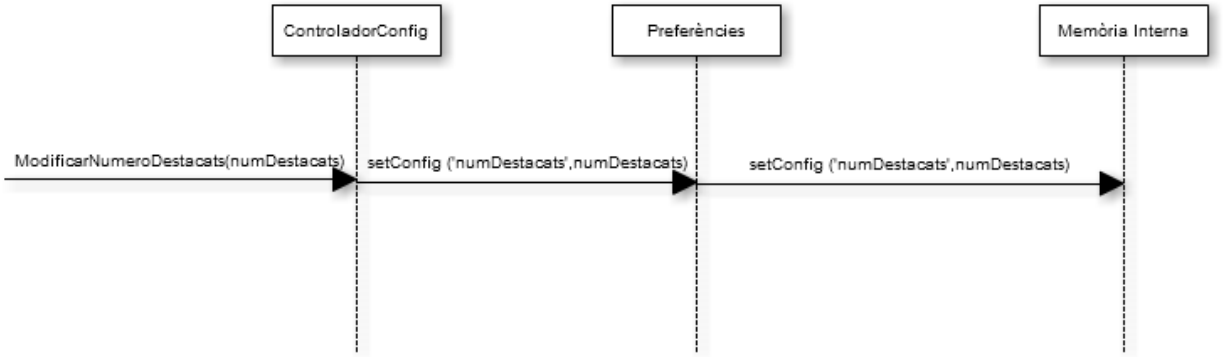
Modificar Vista Mapa



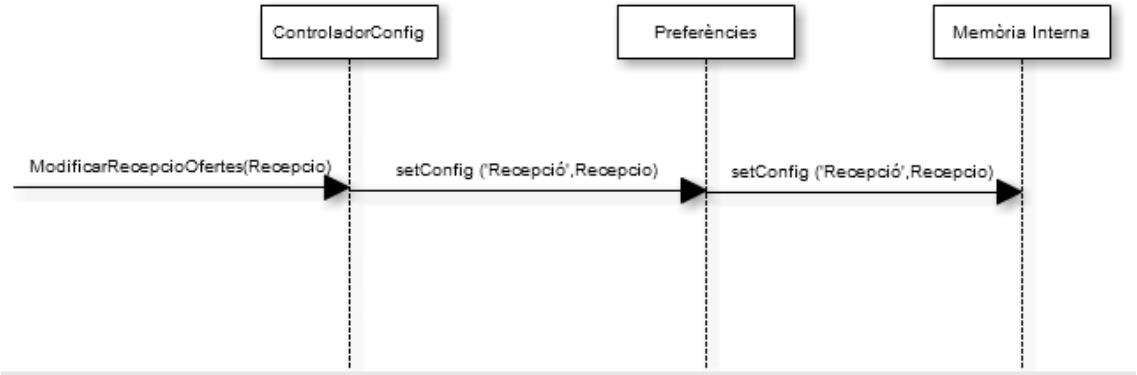
Modificar Radi



Modificar Numero Destacats

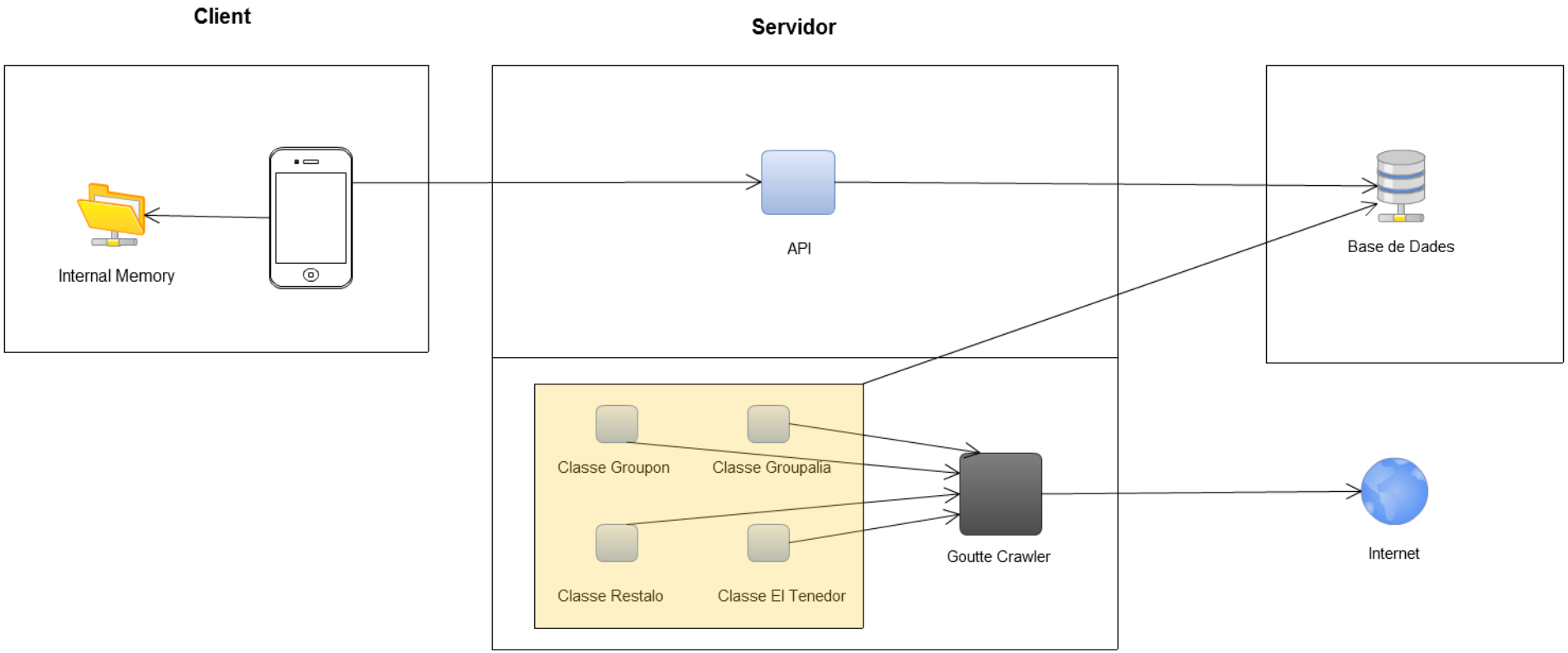


Modificar Recepció Ofertes



Una vegada mostrats els diagrames de seqüència concrets dels casos d'ús, obtinguts a partir del diagrama de classes que s'ha mostrat abans, podem mostrar el diagrama d'arquitectura del sistema, que englobarà no només la part que s'explicarà a l'apartat d'implementació, sinó també els elements que s'han implementat en els estudis previs per a obtenir les ofertes.

Diagrama d'arquitectura



A l'anterior diagrama d'arquitectura podem distingir entre diversos components:

Per una banda, tenim la part del **Client**, on tenim l'**Aplicació** pròpiament dita, de la qual parlarem al següent apartat.

Aquesta aplicació interactuarà amb l'API per a obtenir les dades que es mostraran a l'usuari, per pantalla (les ofertes). Farà crides a unes URL concretes, de les quals parlarem en la fase d'implementació, amb les quals s'accedirà a les diferents funcions disponibles a l'API. L'API retornarà les dades en format JSON[40] i les diverses classes de l'aplicació que s'encarreguin d'interactuar amb l'API, hauran de tractar les dades retornades per a poder mostrar-les per pantalla , en un format molt més amigable.

També destaco en la part del Client la **Internal Memory** . Amb això em refereixo al fet que l'aplicació guardarà informació en la memòria interna del dispositiu. Bàsicament, la informació que guardarem en aquesta memòria serà la configuració de l'aplicació, és a dir, les preferències de l'usuari (quedarà més clar quan s'expliquin les pantalles de l'aplicació).

A la part del **Servidor**, podem distingir dues parts clares. Per una banda, hi ha l'API. Aquesta API serà una classe que s'encarregarà de gestionar les peticions de dades que farà l'aplicació contínuament. El que farà serà fer les queries[41] necessàries a la **base de dades** per tal de tornar el conjunt de dades que l'aplicació ha demanat. Per a accedir a l'API, l'aplicació farà crides a unes URL que el que faran serà indicar a la classe, quina funció de les seves executar, i amb els paràmetres que siguin necessaris. Com hem esmentat abans, les dades seran retornades en format JSON. L'elecció de JSON com a format per a retornar les dades, per damunt del XML, es basa principalment en el fet que és més fàcil de manipular, viatgen menys dades a cada petició i, en general, és un format que potència l'eficiència de l'intercanvi de dades. L'API tindrà un apartat propi, on s'especificarà cada operació individualment.

Per altra banda, al Servidor també tenim el **procés automatitzat d'obtenció d'ofertes** del qual hem parlat abans. El servidor té especificades unes hores en concret, on ha d'executar una sèrie d'accions, per a obtenir o actualitzar les ofertes de la base de dades. Per a obtenir aquestes ofertes, es té una Classe per cada font d'ofertes que utilitzem, que en el cas de El Tenedor, Groupalia i Groupon, el que fan és utilitzar el **Web Crawler** per a extreure la informació, i en el cas de Restalo, es fan connexions a la seva API per a obtenir les seves promocions i ofertes. Aquest procés es pot veure en el diagrama, amb les comunicacions del Crawler i de la classe de Restalo amb **Internet**. Finalment, aquestes Classes desen la informació a la Base de dades, informació que serà explotada per l'API posteriorment. Aquests són els components del sistema i les seves interaccions. A continuació ens centrarem en l'aspecte visual que es vol donar a l'aplicació. S'explicaran les pantalles individualment i es mostrarà finalment un mapa de navegació.

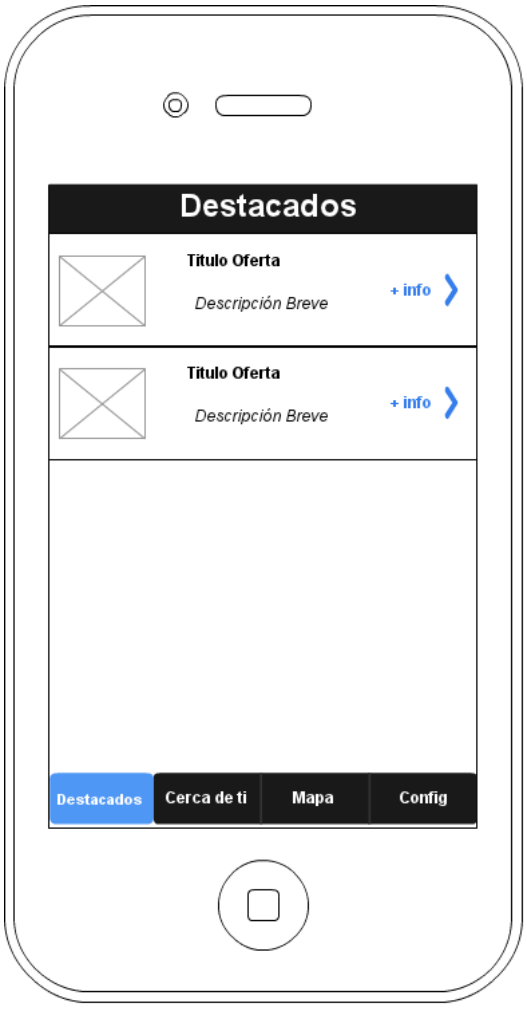
[40] **JSON**: Format lleuger d'intercanvi de dades. Va sorgir per a millorar el rendiment dels intercanvis de dades, que fins al moment es feien en XML, format molt potent però molt pesat.
[41] **Query**: Amb query, em refereixo a una consulta a una base de dades, amb la qual s'obtenen dades

Disseny de l'aplicació

L'aplicació comptarà amb sis vistes, que ara explicarem individualment. Hi ha dos elements que totes les pantalles del sistema tindran en comú, una capçalera on s'indicarà el títol de la pantalla i una barra inferior on hi haurà els botons per a moure's d'un apartat a un altre.

Vista de les ofertes destacades:

Aquesta serà la primera pantalla que es mostrarà una vegada l'aplicació hagi carregat. Podem veure a continuació un disseny esquemàtic del que es vol aconseguir en aquesta pantalla en concret.



Com podem observar al disseny, tenim la capçalera i la barra inferior de què hem parlat que són comunes a tota l'aplicació. Seguint els dissenys típics de les aplicacions per a Iphone, si cliquem damunt un dels botons inferiors, saltarem d'una secció a una altra, i la secció activa en aquell moment és la que està destacada amb un altre color.

Al contingut de la pantalla es mostraran les ofertes destacades del sistema, en forma de llista.

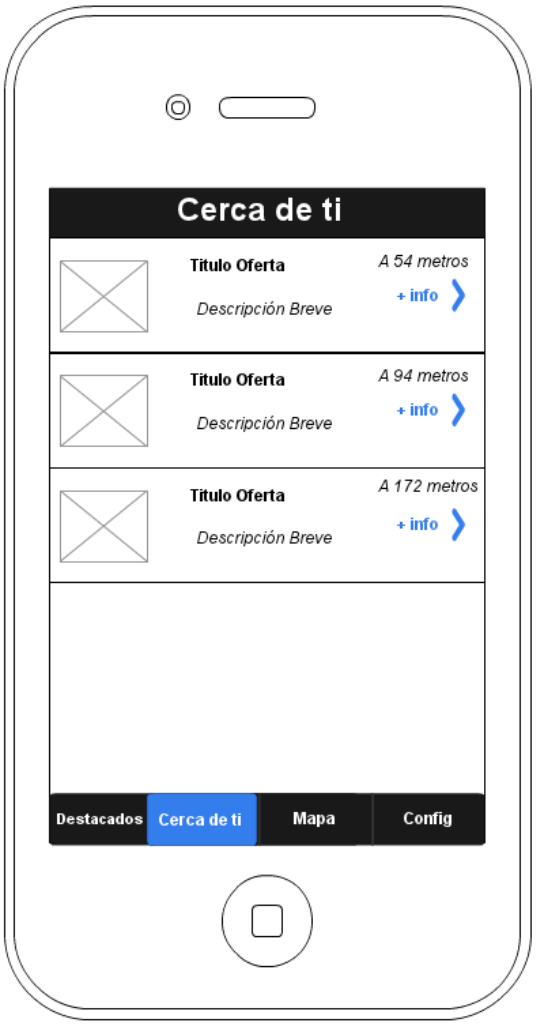
A la llista es mostrarà el títol de l'oferta, una breu descripció i una imatge representativa d'aquesta. Si l'usuari selecciona una de les ofertes, s'anirà a la vista de l'oferta en concret, on es veurà tota la seva informació.

Els criteris que faran que una oferta estigui a l'apartat de les destacades, s'explicaran més endavant. Hores d'ara, ens és suficient pensar que les destacades són simplement les més populars del sistema.

L'elecció d'aquest tipus d'interfície amb una barra superior i una d'inferior per a l'aplicació és principalment a causa del fet que moltes de les aplicacions d'aquest sistema l'han adoptat, i personalment considero que és una manera molt elegant d'organitzar l'aplicació.

Vista de les ofertes del teu voltant

Aquesta pantalla és la que es mostra quan l'usuari ha seleccionat el botó de la barra inferior que conté el text "Cerca de Ti".



Aquesta pantalla és semblant en aspecte a l'anterior, però el funcionament és molt diferent. En aquesta pantalla, es mostraran les ofertes que hi ha a prop de la ubicació de l'usuari. Com podem observar, s'especifiquen les distàncies a la ubicació de les ofertes respecte la posició de l'usuari.

El número d'ofertes que es mostraran en aquesta pantalla depèn de la configuració de l'usuari. Com veurem a la pantalla de Configuració, l'usuari pot definir el seu radi d'acció, és a dir, la distància límit per la qual considerar que una oferta és a prop seu. D'aquesta manera, si un usuari té com a radi d'acció 500 metres, es mostraran només les ofertes que estiguin a 500 o menys metres de la seva posició.

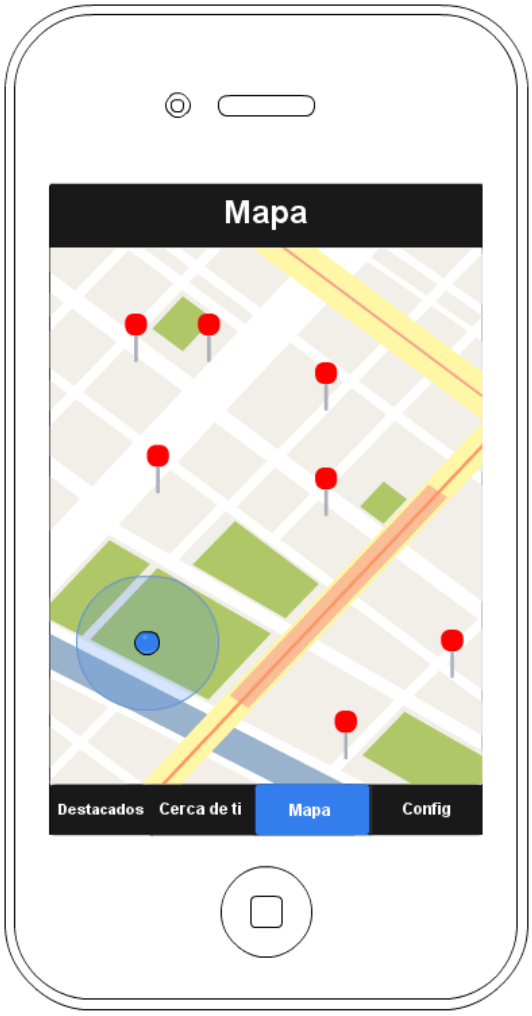
Cada vegada que es cliqui damunt d'aquesta opció, s'actualitzarà el llistat en funció de la posició de l'usuari.

Com a la pantalla anterior, si es clica damunt una de les ofertes, s'anirà a la vista d'aquella oferta en concret (de la qual parlarem més endavant).

Un dels aspectes amb què s'ha de tenir molta cura és en el procés del càlcul de les distàncies. Es considera una part molt important, ja que s'ha de donar a l'usuari una informació fiable. Unes distàncies que no s'ajustin a la realitat faran que l'usuari desconfiï de les dades de l'aplicació i que no la utilitzi.

Vista del mapa de les ofertes

Per arribar a la pantalla que explicarem a continuació s'ha de clicar al botó "Mapa" de la barra inferior de l'aplicació.



En aquesta part de l'aplicació, es veurà un mapa en el qual per un costat es mostrarà la posició de l'usuari i per l'altre es mostraran les ofertes que té al seu voltant en forma de Markers (com els que hem utilitzat en els estudis previs per a col·locar els Tweets capturats a la seva posició geogràfica).

L'usuari ha de ser capaç de treure zoom en aquesta vista, i que es mostrin més ofertes.

Com veurem en l'apartat de la configuració, l'aspecte visual del mapa es podrà canviar (des d'aquella pantalla), mostrant o bé el mapa o la versió amb imatges per satèl·lit, o bé una versió híbrida, visualitzacions típiques que ofereix Google Maps.

S'ha de donar a l'usuari informació sobre les ofertes que estan col·locades al seu voltant en forma de Markers. Si l'usuari selecciona un dels Markers, se l'haurà de portar a la vista d'aquella oferta en concret.

Vista de Configuració

Ara es mostrarà la pantalla de configuració, a la qual s'accedeix clicant damunt l'últim botó de la barra inferior de l'aplicació.

Aquesta pantalla és de les més importants de l'aplicació, ja que permet que l'usuari indiqui les seves preferències i ajusti la configuració de l'aplicació als seus interessos.

Com podem veure, pot ajustar la distància màxima de recepció, que és bàsicament el radi d'acció de què hem parlat abans, el llindar que indica si una oferta és a prop de l'usuari o no (és a dir, si una oferta és a una distància menor o igual a aquest llindar, es mostrarà a la llista de la pantalla de les ofertes properes).

També es permet desactivar la recepció d'ofertes, amb la qual cosa l'aplicació no farà peticions al servidor.

La visualització del mapa també es podrà configurar, i es permetran tres tipus de visualitzacions.

Per últim, també es permetrà augmentar o disminuir el número d'ofertes que es vol que es mostrin a l'apartat de les ofertes destacades.

Aquesta serà l'única pantalla que guardarà informació en la memòria interna del dispositiu, on hi haurà la configuració que l'usuari ha establert i que s'aplicarà al comportament de tota l'aplicació. S'ha de destacar que d'aquesta pantalla l'opció més important és la d'editar el radi d'acció. Opció que realment es considera clau perquè l'usuari tingui la sensació de poder personalitzar el comportament de l'aplicació.

Ara es mostraran dues pantalles a les quals no es pot accedir directament amb el menú inferior.

Vista de visualització d'una oferta

Per arribar a aquesta pantalla, s'ha de clicar damunt d'una oferta, ja sigui a la llista de les destacades, a les que tens al voltant, o bé quan cliques damunt d'un dels Markers de la vista de les ofertes al Mapa.

Podem observar dues peculiaritats en aquesta vista. La primera és que no hi ha cap dels botons inferiors seleccionats. Això es produeix pel fet que es pot accedir a aquesta vista des de diverses pantalles diferents, la qual cosa fa que marcar-ne una no tingui sentit.

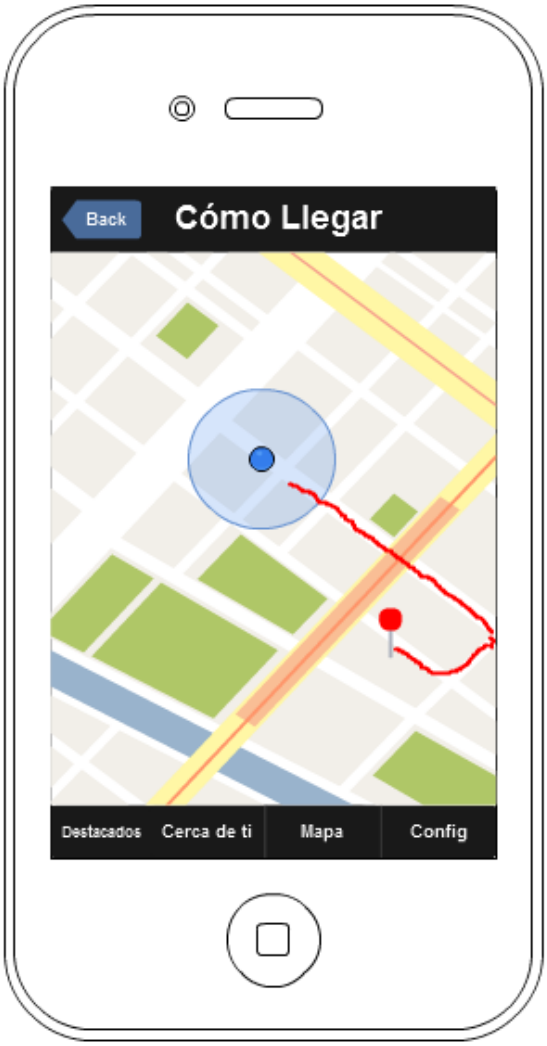
L'altra peculiaritat és el botó de "back" a la capçalera. Aquest botó el que farà serà simplement tornar a la vista anterior.

Al contingut tenim la imatge de l'oferta, el títol, la distància de l'usuari a l'oferta i la seva descripció. A més, veiem que hi ha un botó gran que té com a text "Cómo Llegar". Això ens portarà a una vista que explicaré a continuació, on bàsicament, se'ns marca una ruta al mapa, per arribar des de la nostra posició actual, a l'oferta en concret.

Amb aquesta pantalla, el principal objectiu és que l'usuari pugui veure la informació completa d'una de les ofertes, d'una manera clara i ordenada.

Vista de com arribar a una oferta

Per accedir a aquesta pantalla, s'haurà de clicar al botó que hem vist a la pantalla anterior, anomenat "Cómo Llegar".



Podem observar que en aquesta pantalla també tenim un botó de "back", que ens enviarà de tornada a la pantalla de visualització de l'oferta seleccionada.

En aquesta pantalla es mostrarà a l'usuari la seva posició, la de l'oferta i una ruta proposada per a arribar a l'oferta en concret.

Considerem això com una funcionalitat molt interessant, ja que no només informem de l'existència d'ofertes al voltant de l'usuari, sinó que els guiem perquè puguin fer-les efectives i es puguin beneficiar dels descomptes, promocions o ofertes que oferim.

Una vegada explicades individualment totes les pantalles, mostrarem un mapa de navegació amb el qual es pot veure d'una manera esquemàtica i resumida tot el que hem explicat de com es relacionen les pantalles de l'aplicació. Les fletxes als botons de la barra inferior indiquen que clicant en aquells botons, es va a la pantalla que es mostra.

Mapa de Navegació

Riscos

A continuació es farà una anàlisi dels principals riscos que es poden materialitzar en el desenvolupament del sistema. Per a cadascun dels riscos, es posarà la probabilitat que es materialitzin i l'impacte que podrien tenir. A més es proposaran mitigacions (mesures per a evitar que es materialitzi el risc) i contingències (mesures preses una vegada s'ha materialitzat un risc per a minimitzar l'impacte del mateix).

Falta de temps

Probabilitat: Alta
Impacte: Alt

Aquest és probablement el risc que més em preocupa. El temps de què es disposa per a desenvolupar l'aplicació és molt just i s'haurà d'aprendre un llenguatge i una filosofia de programació totalment nova.

Com a **mitigació** es té que la planificació s'ha fet tenint en compte aquests factors i que s'ha establert un marge d'adaptació al nou sistema, suficient, com per a acabar en el temps establert. A més es disposen cursos en línia i bibliografia que es consultarà abans de començar, per a comptar amb una orientació bàsica.

Com a **contingència** ens queda com a única opció retallar funcionalitats. En cas que fos necessari, tot requeriment opcional serà descartat i la funcionalitat del càlcul de rutes entre usuari i oferta seria la que es descartaria per tal de poder acabar a temps.

Interfície no del tot intuïtiva

Probabilitat: Baixa
Impacte: Mitjà

Aquest no és un risc massa preocupant, ja que en tot moment l'estratègia que plantejaré com a mitigació farà que sigui molt poc probable que es materialitzi aquest risc.

Com a **mitigació**, el que es farà serà dissenyar la interfície i demanar a persones que no tinguin res a veure amb el projecte si ho veuen intuïtiu.
En tot moment s'escoltarà les seves opinions i es faran els canvis pertinents per tal de fer que la interfície sigui el més intuïtiva possible.

En aquest cas, no considero necessària una **contingència**, ja que el procés de mitigació eliminarà la possibilitat que el risc s'acabi materialitzant.

Temps de càrrega d'ofertes alt

Probabilitat: Mitjà
Impacte: Alt

Amb això em refereixo al fet que l'obtenció de les ofertes per part de l'aplicació sigui molt lent, cosa que farà que l'usuari tingui temps d'espera massa alts.

Com a **mitigació**, s'intentarà que el número de connexions amb el servidor siguin les mínimes, i que les dades que s'intercanviïn siguin també les menys possibles.

Com a **contingència**, es limitarien les dades que l'usuari podria rebre, per tal que l'aplicació fos més fluïda i tingués menys temps d'espera.

Dificultats d'aprenentatge

Probabilitat: Baixa
Impacte: Alt

Això és un risc que té a veure amb la meva capacitat d'aprenentatge. Es té el temps just per a aprendre a utilitzar Objective-C, Cocoa Touch i per a adaptar-se l'entorn de desenvolupament dels ordinadors d'Apple (això es parlarà més en profunditat a l'apartat d'implementació) . Tot i que considero que tinc una bona capacitat d'aprenentatge, de no ser així, es perdria un temps excessiu, que faria que no s'acabés el projecte en els terminis establerts. Tot i així, la probabilitat la considero baixa, donat els meus coneixements de llenguatges similars als que s'hauran d'aprendre.

Com a **mitigació**, s'accedirà el més aviat possible a tot el material disponible a la xarxa sobre el tema i s'intentarà aprendre els principals aspectes teòrics el més prest possible.

No hi ha **contingència** possible.

Especificació de l'API

Com s'ha comentat anteriorment, per a gestionar la interacció amb la base de dades, tindrem una API, que no serà més que una classe que ens servirà per a accedir a la informació que necessitem en cada moment a l'aplicació.

La nostra API funcionarà d'una manera similar a les API que tenen serveis com poden ser Google Books, Twitter o Facebook. El que farà serà fer crides a una URL en concret, que variarà segons l'operació que vulguem que s'executi i retornarà les dades demanades en un format determinat, que l'aplicació utilitzarà per a construir el contingut de la interfície.

Les crides es faran a unes URL que tindran un aspecte similar a aquest:
- http://*nomdeldomini*/API/api.php?q=*"operacio-a-realitzar"*&*nomparam1="param1val"* …

El que indicarem amb aquest tipus d'URL és que s'accedirà a una carpeta en concret del servidor (/API), en aquesta carpeta, s'executarà un mòdul escrit en php anomenat API, que rebrà com a paràmetres l'operació que cal realitzar (la "q") i els seus paràmetres. Aquest mòdul s'encarregarà de instanciar a la classe que realment tindrà els mètodes demanats i de cridar a les seves funcions pertinents.

Aquesta classe serà la que a continuació s'especificarà. Es mostraran els contractes de les principals operacions que podrà executar aquesta classe.

Classe API

Ara es mostraran els contractes de les operacions de l'API. S'ha de comentar que primer es mostren una sèrie de funcions marcades com a bàsiques, i que després es mostren unes altres marcades com a opcionals. Això és perquè aquestes funcions només seran implementades si es disposa del temps necessari per a fer-ho. És a dir, són les funcions menys prioritàries de l'API i que no influeixen en el funcionament bàsic de l'aplicació.

Funcionalitats Bàsiques

Public Function ObtenirDestacats (**int** numeroDestacats): **Returns** (**String** OfertesDestacades)

Pre: Es rep el número d'ofertes destacades que l'usuari ha especificat que vol veure. És un número superior a zero i inferior al màxim permès a la configuració.
Post: Es retornen tantes ofertes destacades del sistema com s'hagin especificat en el paràmetre. Si al sistema no hi ha ofertes, es retornarà un String buit. Per a definir quines ofertes són més destacades, ens fixarem en un camp anomenat "score" de la base de dades, que no és més que la valoració que tenien les ofertes en les fonts originals, normalitzades per a tenir-les en una escala de puntuació igual. Es retornaran les "n" ofertes que hagi demanat l'usuari ordenades de manera decreixent segons la seva puntuació.

Public Function ObtenirProperes (**float** latitud, **float** longitud, **int** Radi)
: **Returns** (**String** OfertesProperes)

Pre: Es reben la latitud i la longitud de l'usuari, en un format correcte, a més d'un Radi en metres, que serà superior a zero i inferior al màxim establert en la configuració de l'aplicació.
Post: Retorna les ofertes de la base de dades tals que la distància entre la posició de l'usuari i la posició de l'oferta, sigui inferior al Radi que s'ha passat com a tercer paràmetre.

Public Function ObtenirPerMapa (**float** latitudC, **float** longitudC, **int** RadiMap): **Returns** (**String** Ofertes)

Pre: Es reben la latitud i longitud d'on l'usuari té el mapa centrat, a més del radi de visió que es té al voltant d'aquest centre. Això dependrà del Zoom del mapa, com més zoom, més petit serà aquest radi de visió, com menys, més gran serà.
Post: Es retornen totes les ofertes que hi ha dins la circumferència que es forma des del centre que especifiquen els dos primers paràmetres i amb el radi especificat.

Public Function ObtenirOferta(**int** idOferta): **Returns** (**String** Oferta)

Pre: Arriba una Id d'una oferta en concret, correcta, és a dir, existent a la Base de Dades.
Post: Es retorna la informació completa d'una oferta en concret.

Funcionalitats Opcionals

Public Function ObtenirOfertesSource (**String** Source): **Returns** (**String** OfertesSource)

Pre: S'obté com a paràmetre un dels proveïdors d'ofertes que es tenen a la Base de dades.
Post: Es retornen totes les ofertes que pertanyen a aquell proveïdor.

Public Function FerReserva (**int** IdRestaurant, **String** mail, **int** num ,**String** Nom, **Time** Hora,
Date dia) : **Returns** (**Boolean** status)

Pre: Es passa la Id correcta d'un restaurant de Restalo, una adreça de correu electrònic, un número de comensals, un nom per a la reserva, una data i una hora. Totes les dades estaran en el format esperat.
Post: Es fa una reserva al restaurant indicat de Restalo, utilitzant les funcionalitats de la seva API, en el dia, hora i per als comensals que hem indicat.

Implementació

En aquest apartat de la memòria el que s'explicarà serà la part més tècnica de la implementació de l'aplicació mòbil que hem estat dissenyant fins ara.

Inicialment, parlarem sobre com programarem l'aplicació. Explicarem a continuació els fonaments bàsics de com es programen aplicacions per iOS.
Per començar, parlarem sobre l'entorn sobre el qual es desenvoluparà tot el projecte.

Entorn de programació

Com bé sabem, per a desenvolupar una aplicació per a dispositius Apple, s'ha d'implementar des d'un dispositiu de la mateixa Apple. Per això, s'ha programat amb un Mac Mini, sobre el sistema operatiu MacOS X Lion. Les proves s'han fet utilitzant un Iphone 3Gs, per tal de veure la fluïdesa de l'aplicació en un dispositiu molt menys potent que els models actuals i, d'aquesta manera, estar segurs que estem programant l'aplicació d'una manera eficient.

Quant al programari utilitzat per al desenvolupament, hem de parlar de dues grans eines. Per una banda tenim un IDE com és Xcode, i per altra, un software per a dissenyar interfícies, anomenat Interface Builder. Actualment, el segon està integrat dins el primer, però fins fa relativament poc, eren dos programes totalment independents que es comunicaven entre ells.

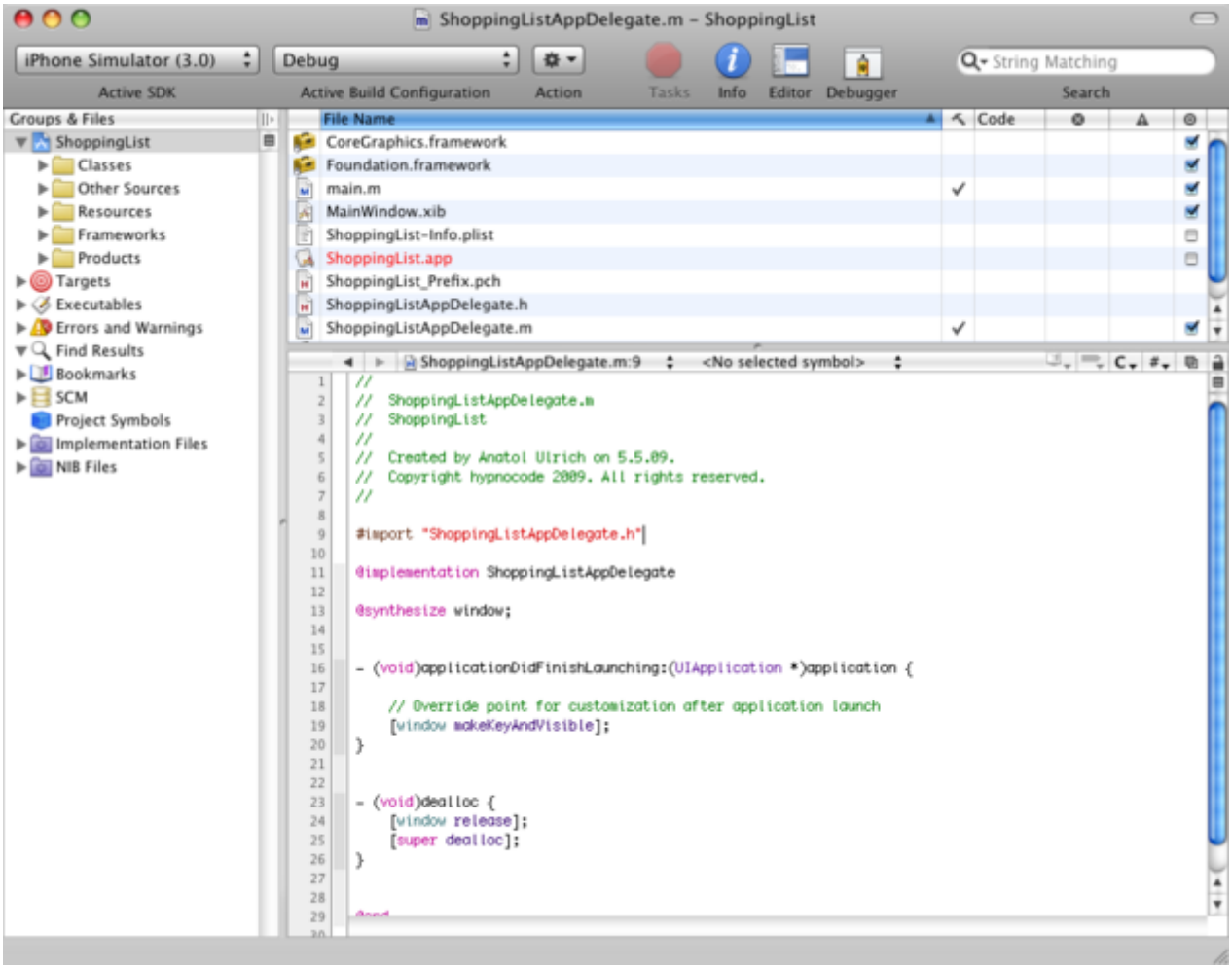
Xcode i Interface Builder

Xcode és un entorn de desenvolupament, elaborat per Apple, i que es pot descarregar gratuïtament si ets usuari de Mac. Es tracta d'una eina molt potent que permet compilar codi de molts llenguatges de programació (C, C++, Java, AppleScript, i sobretot el que ens interessa, Objective-C). S'ha utilitzat la versió 4.3 en el nostre cas. Com hem comentat abans, només funciona sobre sistemes operatius MacOS, i amb el que sí és compatible, és amb gran quantitat de sistemes de control de versions, com poden ser Subversion, Git o CVS.

Permet compilar i executar les aplicacions programades i, particularment, en el cas de les App d' iOS, les pot executar utilitzant un emulador de la família de dispositiu a les quals es vol dirigir l'aplicació (Iphone o iPad). És una eina en general molt potent a l'hora de programar, ja que té un auto completat molt precís, i tot i tenir gran quantitat de funcionalitats no carrega tant el sistema com altres IDEs com Netbeans.

S'utilitzarà per a programar en el llenguatge en Objective-C (llenguatge de què parlarem una mica més endavant), amb l'ajuda del Framework Cocoa Touch (del que també parlarem). Tot i que no s'ha dit explícitament, aquest IDE porta el SDK de iOS incorporat, convertint-se en el entorn de desenvolupament ideal per a iOS.

Aquí podem observar una captura de pantalla, de l'aspecte que té l'Xcode:



Com veiem, té moltes opcions que ens ajudaran en el desenvolupament. A l'esquerra tenim una llista on podem recórrer l'estructura del nostre projecte, entre d'altres coses. A la vista inferior dreta, podem observar el component d'edició de codi pròpiament dit.

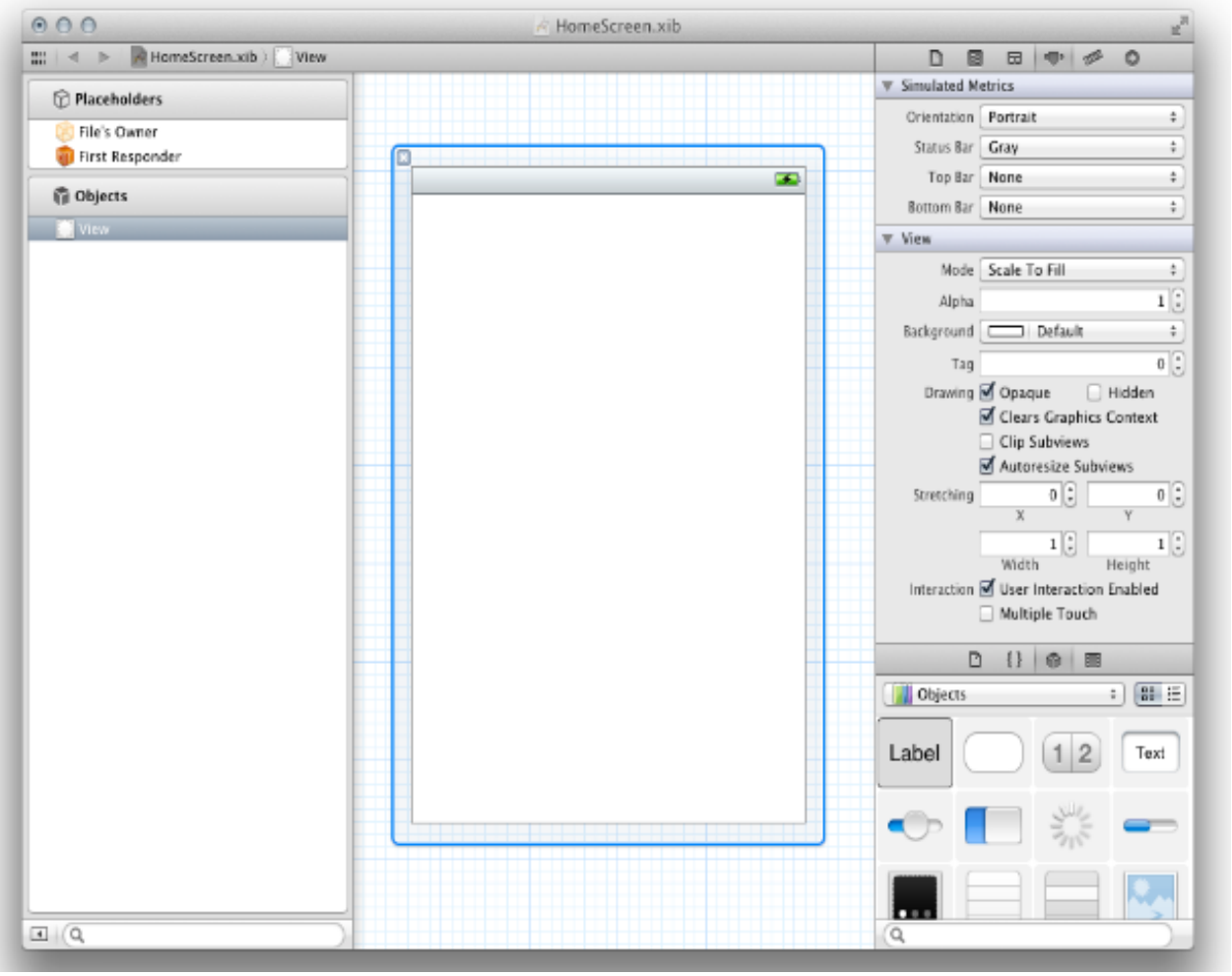
Hem comentat abans que també s'utilitzarà una eina per al desenvolupament de les interfícies. Aquesta actualment està totalment integrada amb l'Xcode i s'anomena Interface Builder.

L'Interface Builder és una eina destinada a dissenyar interfícies de manera senzilla. Es poden afegir els diversos components que hi ha disponibles, amb la filosofia "drag & drop", cosa que simplifica la feina al programador.

Els arxius que maneja aquest programa tenen l'extensió .xib, i dins un projecte d'Xcode, hi haurà tantes vistes com arxius .xib.

Fent doble click damunt un d'aquests fitxers, se'ns obrirà l'Interface Builder, i podrem modificar la vista en concret.

Podem veure a continuació una captura, de l'aspecte visual de l'eina:



Com podem veure, a la part central tenim la vista i a la dreta moltes opcions que poden ser editades. Veiem a la part inferior dreta els diversos components que es poden afegir a la vista, cosa que podrem fer-ho simplement agafant els components i arrossegant-los a la vista central del programa.

Ara s'explicaran una mica els llenguatges implicats en el desenvolupament en aquest tipus d'aplicacions.

Objective-C

Objective C és un llenguatge de programació que va néixer a principis dels 80. Podem destacar com a característiques seves que és orientat a objectes i que va ser creat com a superconjunt de C.

És a dir, pel fet de ser un superconjunt, es pot compilar qualsevol programa escrit en C amb un compilador d'Objective C, a més de poder incloure lliurement codi C dins d'una classe d'Objective C.

Com a característica diferenciadora d'Objective C, tenim que el tipat de les variables pot ser tant estàtic com dinàmic. Es pot posar com a retorn d'una funció, per exemple, un objecte d'una classe que anomenen "Id", el qual ens indica que aquella funció pot retornar un element de qualsevol classe.
Evidentment, això és opcional i l'usuari pot utilitzar o no aquesta característica del llenguatge.

Aquest llenguatge es combina amb un Framework anomenat Cocoa Touch, que s'encarrega de proporcionar una capa d'abstracció sobre el sistema operatiu iOS i que es puguin utilitzar les funcionalitats del sistema, d'una manera senzilla des d'Objective C. Si no s'utilitzés, els desenvolupadors tindrien un llenguatge de programació compilat similar a C/C++ o Java, amb el qual podrien fer aplicacions convencionals.

Per a fer-nos una idea de quin aspecte té el codi, a continuació es mostrarà el codi d'una classe que representa fraccions. El que hem de destacar inicialment, és que tal i com passa a C o C++, una classe la forma un fitxer .h (de header, com als llenguatges esmentats) i un fitxer .m (que en el cas de C, seria .c, i en C++ .cpp, on hi ha la implementació de la classe).

Arxiu .h

```
#import <Foundation/NSObject.h>

@interface Fraction: NSObject {
    int numerator;
    int denominator;
}

-(void) print;
-(void) setNumerator: (int) n;
-(void) setDenominator: (int) d;
-(int) numerator;
-(int) denominator;
@end
```

Veiem que al fitxer hi ha les funcions i accions de la classe i els atributs (en aquest cas, són a un apartat anomenat @interface). El signe negatiu d'abans dels mètodes indica que són a nivell d'instància.

En el cas de tenir una funció estàtica, és a dir, una funció que actua a nivell de classe, aquesta tindria un signe "+" al davant.

Arxiu.m

```
#import "Fraction.h"
#import <stdio.h>

@implementation Fraction
-(void) print {
   printf( "%i/%i", numerator, denominator );
}

-(void) setNumerator: (int) n {
   numerator = n;
}

-(void) setDenominator: (int) d {
   denominator = d;
}

-(int) denominator {
   return denominator;
}

-(int) numerator {
   return numerator;
}
@end
```

I aquí tenim la implementació. Veiem que el codi és molt senzill. Podem destacar el fet que s'hagi fet un print per pantalla, utilitzant el típic "printf" de C.

En pàgines properes, ja es parlarà de les classes i arxius d'interfície de la nostra aplicació.

Implementació del projecte: Aplicació mòbil

Abans d'entrar a comentar aspectes importants de com s'ha desenvolupat l'aplicació, considero interessant fer una introducció, perquè la terminologia que s'utilitzarà quedi clara, i perquè en tot moment es tingui clar què és el que es vol dir. Parlarem sobre diferents aspectes relacionats amb el desenvolupament, com són la tipologia de fitxers involucrats, sobre com es desenvolupen els projectes utilitzant l'IDE, i alguns conceptes que potser ens semblen una mica diferents respecte a la programació més tradicional, entre d'altres coses. Una vegada s'hagin explicat aquests aspectes, passarem a explicar com s'ha implementat el codi i què fa exactament cada un dels fitxers que tenim al projecte.

Tipus de Fitxers

L'aplicació que desenvoluparem, per a dispositius amb iOS, comptarà amb una sèrie de classes i fitxers de diferents tipus. Ara explicarem breument els diversos tipus de fitxers amb què ens trobarem i dels quals parlarem:

- **Fitxers .h**: D'aquests fitxers, ja n'hem parlat breument. Són els fitxers de capçalera, en què es trobaran les declaracions de les propietats d'una classe i les capçaleres dels mètodes, entre d'altres coses.

- **Fitxers .m**: En aquests fitxers, es trobaran les implementacions de les capçaleres de funcions que teníem al seu respectiu fitxer .h . Aquí serà on es trobarà el codi de les nostres classes.

- **Fitxers .xib**: Aquest tipus de fitxers seran els que editarem amb l'Interface Builder i seran els que contindran la part visual de l'aplicació. Hi haurà un fitxer .xib per a cada vista de la nostra aplicació.

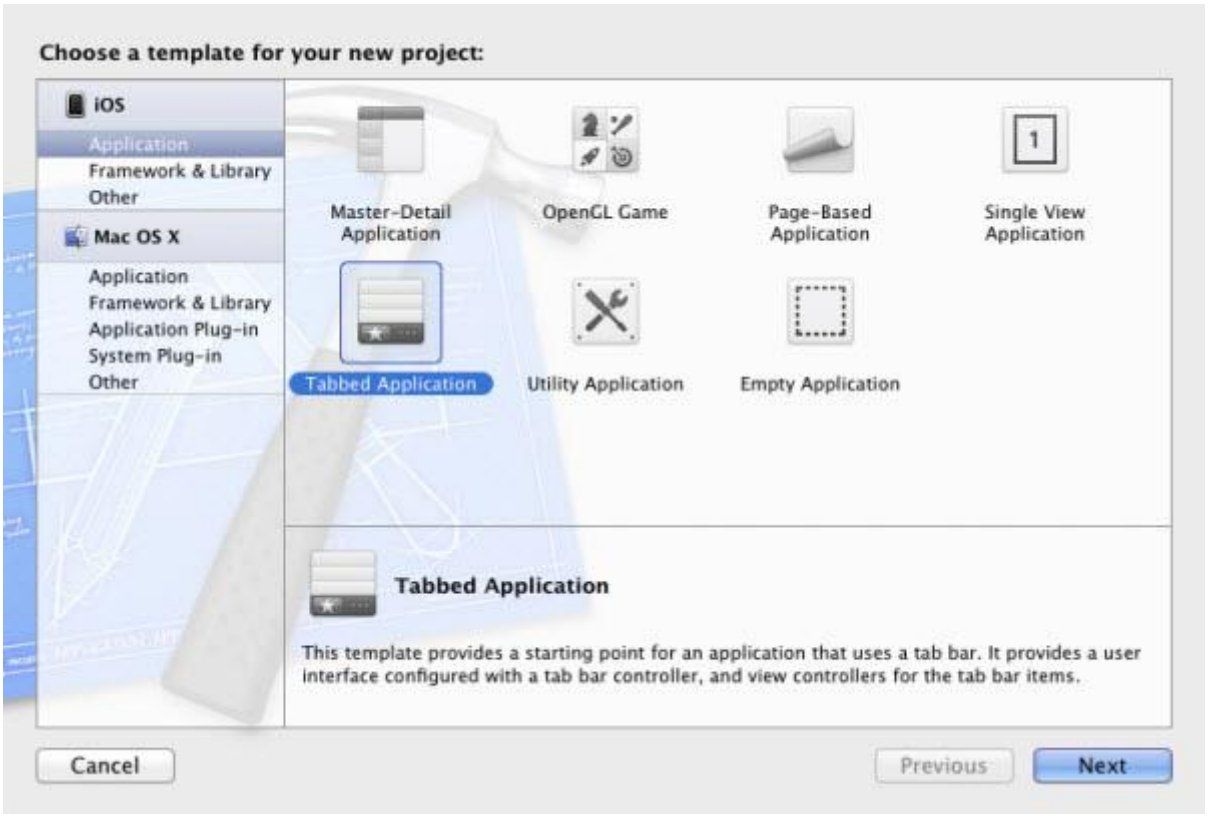
- **Imatges**: Es tindrà una sèrie d'imatges al projecte, que s'utilitzaran per a donar un aspecte més amigable al sistema.

Hi ha més tipus de fitxers al projecte que et genera l'Xcode quan es crea un projecte per a iOS (com els fitxers .plist), però no els esmentarem, ja que són fitxers que es generen automàticament i que no haurem de modificar.

Creació del projecte

Com hem vist prèviament al disseny de la interfície, l'aplicació estarà composta per quatre pestanyes o botons a una barra inferior, que marcaran les quatre vistes principals de l'aplicació.

A l'hora de triar quin tipus de projecte crear, l'Xcode ens dóna diverses opcions. A continuació podem veure-ho:



Com s'observa, hi ha l'opció de crear un projecte, utilitzant la plantilla "Tabbed Application", que el que ens generarà serà l'esquelet del codi per a programar una aplicació d'aquest tipus, la qual cosa facilita el nostre treball.
Ens generarà una plantilla buida, amb una interfície que tindrà dues pestanyes i un contingut predeterminat. Veurem una interfície similar a la següent:

A nivell de codi, el que ens ha generat són dos fitxers .xib, el FirstView.xib i el SecondView.xib, que seran els dos fitxers d'interfície, que en aquest cas només estaran formats per un Label amb un text (Screen One en el cas que veiem).

També ens haurà generat un FirstView.h, un FirstView.m i el mateix amb la SecondView. Aquestes classes seran els controladors que gestionaran la seva vista en concret. El que s'haurà de fer en el nostre cas, és afegir dues vistes més i generar tant els 2 .xib, com els fitxers de classe corresponents.

Una vegada fet això, ja es tindrà una estructura amb quatre pestanyes. A partir d'aquest punt, ja s'hauran d'afegir els components utilitzant l'Interface Builder i gestionant el codi a les seves classes corresponents.

Per tant, podem concloure que tindrem una classe per a cada pestanya, que s'encarregarà de gestionar la seva vista corresponent. Aquestes classes tenen com a particularitat que hi ha una sèrie de mètodes que es poden implementar i que s'executen automàticament quan es compleixen certes condicions. Els exemples que més s'utilitzaran són els següents:

- **ViewDidLoad**: Com indica el seu nom, aquesta funció s'executarà quan s'hagi carregat la vista. Implementant aquesta funció, podem personalitzar la inicialització de la vista. No hem de confondre el concepte de carregar la vista, amb el fet que la vista aparegui. Aquest mètode només s'executa la primera vegada que carrega la vista.

- **ViewDidUnload**: Aquest és el mètode invers a l'anterior, s'executarà quan la vista es destrueixi (generalment quan tanquem l'aplicació). És útil per si volem efectuar alguna acció abans de tancar l'aplicació (com alliberar recursos o enregistrar alguna dada).

- **ViewDidAppear**: Aquest serà el mètode que més freqüentment editarem, ja que s'executa cada vegada que la vista apareix, és a dir, quan seleccionem la pestanya corresponent. Serà molt útil, per exemple, per mostrar el Mapa o per fer una crida al servidor, demanant les ofertes properes, segons la posició de l'usuari.

- **ViewDidDissappear**: Tal i com ha passat amb el Load i l'Unload, aquest és el mètode invers a l'anterior, s'executa quan seleccionem una pestanya que no és la que representa la classe. Aquest mètode serà especialment útil per netejar elements de la interfície.

Tal i com hem vist, aquests quatre mètodes s'executen automàticament quan es compleixen certes condicions. Això serà un aspecte nou, respecte a la programació tradicional que ens anirem trobant en el desenvolupament del projecte.

Interacció entre Xcode i Interface Builder

Anteriorment s'han introduït les dues eines principals que s'utilitzaran. Per una banda tenim l'Xcode, que és l'IDE, amb el qual editarem el codi de les classes del nostre sistema (fitxers .h i .m). Per altra banda, tenim l'Interface Builder, el programa amb el qual d'una manera senzilla (drag & drop), crearem les vistes per a l'aplicació (els fitxers .xib). Però clar, les classes han de poder accedir als elements de la interfície, per a poder modificar els seus valors, afegir nous elements, treure'n … En definitiva, han de poder interactuar.

Per a fer això haurem de recórrer al concepte **IBOutlet i IBAction**.

Els **outlets** podrien ser definits com les representacions d'elements de la interfície, en els seus controladors. D'aquesta manera, si a la interfície tenim un Label i volem modificar el seu valor, haurem d'especificar amb l'Interface Builder que aquell element és un outlet d'aquell controlador. Aquesta acció el que provoca és que, al fitxer .h del nostre controlador, aparegui una propietat nova, amb aquest aspecte:

```
@property IBOutlet UILabel *myLabel;
```

Això indica que myLabel és un atribut més del nostre controlador. A partir d'ara, si volem canviar el text, per exemple d'aquest label, des del codi del nostre controlador ho podem fer d'una manera tant senzilla com aquesta:

```
self.myLabel.text = @"Hello World!!";
```

Una **IBAction** és una funció que s'executarà quan l'usuari hagi fet un canvi en un element de la interfície. Per exemple, si tenim un UISlider com aquest:

Ens pot interessar que quan es faci un canvi en aquest element (arrossegant la bola que a la imatge està bé al mig), això internament executi algun tipus d'acció. Per a fer això, el que s'haurà de fer és amb l'Interface Builder indicar que aquest component, si fa saltar l'event ValueChanged (característic d'aquest component), executi l'IBAction que nosaltres indiquem.

Per exemple, podem tenir una IBAction com la següent:

```
-(IBAction) laMevaIBAction:(UISlider *)sender{

    NSLog(@"El Valor del Slider es igual a : %f", sender.value);
}
```

Amb aquesta acció, aconseguirem que un canvi a l'Slider provoqui un print del seu valor a la consola del sistema.

El paràmetre que rep l'IBAction sempre és l'element amb el qual s'ha interactuat per a fer saltar l'IBAction. Això és molt útil per a accedir a les característiques del component, com veiem a l'exemple, obtenim el seu valor.

Sintaxi Especial

L'Objective C té diverses particularitats, entre les quals hi ha la seva particular sintaxi. A continuació, es mostrarà algun exemple d'això de manera il·lustrativa.

```
oferta *of = [[oferta alloc]init];
```

Aquesta és la manera que té Objective C d'inicialitzar una instància d'una classe. En aquest cas, s'està creant un objecte de la classe oferta. Això és equivalent al tradicional:

```
oferta of = new oferta();
```

Els tradicionals valors dels booleans, "true" i "false", en aquest context han estat substituïts per "YES" i "NO". Per a cridar a una funció d'una instància, utilitzarem aquesta sintaxi:

```
[data JSONValue]
```

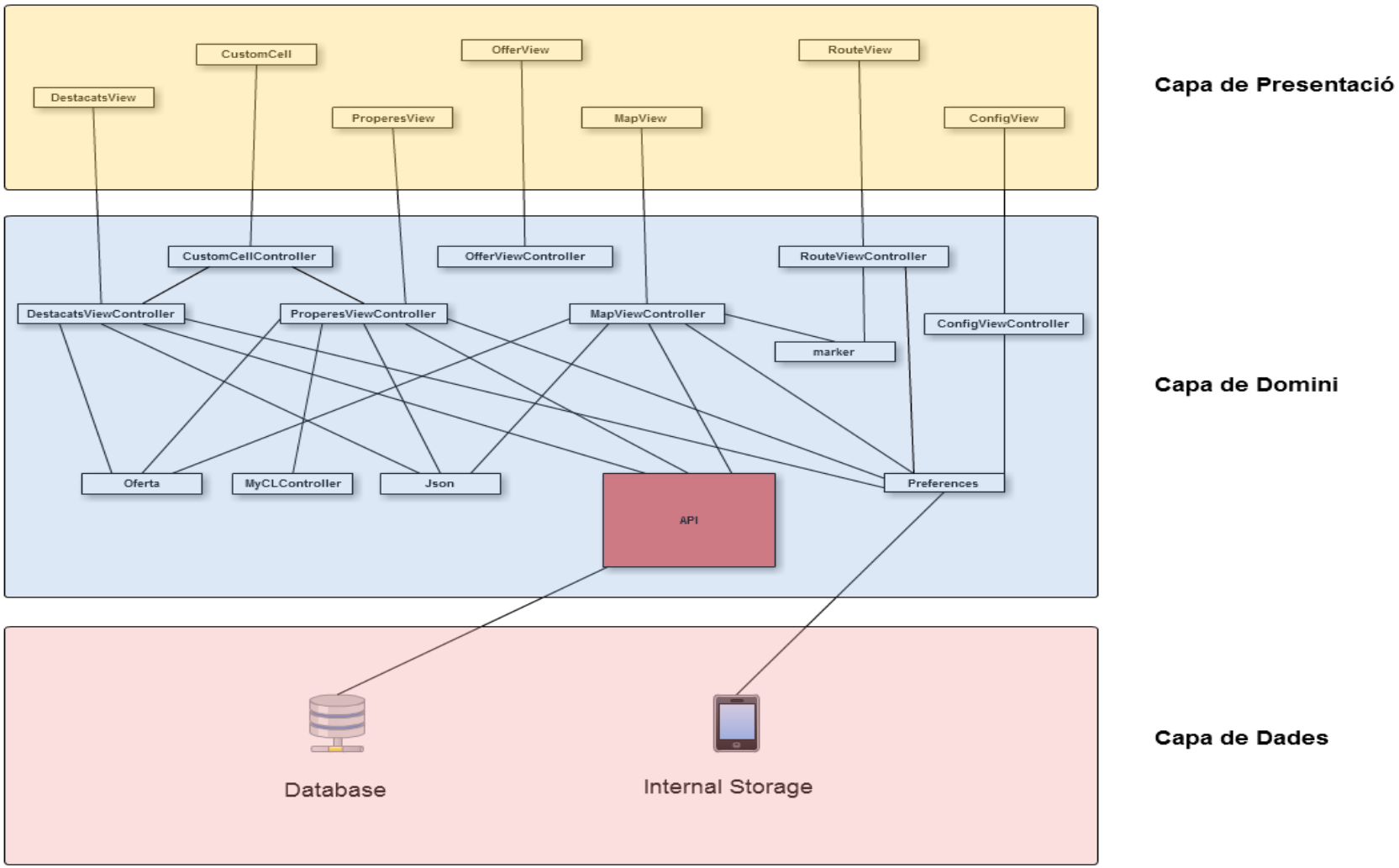
En aquest punt, el que hem fet és aplicar la funció JSONValue a l'objecte "data". Si a una funció li passem un paràmetre, ho farem d'aquesta manera:

```
[offer setNom:@"Joan"]];
```

A l'objecte offer, se li aplicarà la funció setNom, que setejarà el nom que passem com a paràmetre, que en aquest cas és un String (cosa que s'indica amb el símbol "@" i té com a valor "Joan").

En aquest punt, ens podríem estendre molt, però considero que l'aprenentatge d'Objective C no és l'objectiu d'aquest document. Tot i així, amb aquestes pinzellades, es podrà entendre millor el desenvolupament de l'aplicació.

Diagrama de Classes definitiu



En aquest diagrama, podem observar com interactuen els diversos components del nostre sistema. Es poden veure clarament les tres capes i com aquestes col·laboren. No s'han afegit mètodes ni atributs perquè es pogués veure amb una mica més de claredat.

Podem observar com, per a cada vista, tenim el seu respectiu Controlador. A part, tenim una sèrie de classes que s'utilitzen molt, com poden ser les Preferències, Json o la classe Oferta. També podem observar que la classe API està marcada amb un color diferent. Això és perquè tot i formar part de la capa de domini, no forma part de l'aplicació, ja que és una classe del servidor. Com a detall, es veu clar que les vistes entre elles no interactuen directament, sinó que ho fan a través del seu respectiu controlador.

Podem veure com entre la capa de domini i la de dades només hi ha dos nexes. L'API, que es comunica amb la base de dades, i la classe Preferences, que farà el mateix amb la memòria interna del dispositiu.

Tampoc no s'han afegit les cardinalitats en aquest punt, per a donar més una visió general de com interactuen les classes i que es pogués veure clarament. Tota la informació que no s'ha vist amb l'ajut el diagrama s'especificarà a continuació per a cadascuna de les classes.

Implementació de les Classes

Abans de res, he d'aclarir que és possible que molt del codi s'executi en les funcions de què havíem parlat abans com ViewDidLoad. Per tant, hi haurà classes on aparentment no hi ha mètodes, però realment el codi s'està executant en aquestes funcions.

A continuació, es parlarà en detall de totes les classes que han aparegut al diagrama de classes que s'ha mostrat anteriorment.

Oferta

Descripció General

El propòsit d'aquesta classe és representar el concepte d'"Oferta", amb totes les seves característiques. És una de les classes més utilitzades de l'aplicació.

Atributs

idOferta : La Id de l'oferta dins la nostra base de dades.
lat : Latitud on podem localitzar l'oferta.
lng : Longitud on podem localitzar l'oferta.
idRestalo : La Id de Restalo, en el cas que l'oferta tingui com a font Restalo
nom : El nom que identifica l'oferta.
descripcion : La descripció de la mateixa.
link : Un link cap a la URL de l'oferta.
distancia : Distància de l'usuari a l'oferta.
source : La font de la qual s'ha obtingut l'oferta.

Mètodes

Getters i Setters per a tots els atributs.

Preferences

Descripció general

Aquesta classe serà l'encarregada de gestionar la configuració de l'usuari. Servirà tant per emmagatzemar les preferències en la memòria interna del terminal, com per recuperar-les d'aquesta memòria. Serà l'única classe del sistema que interactuï amb la memòria del terminal.

Mètodes

+ (NSInteger)getOfferReception

Obté el valor de la recepció d'ofertes que està emmagatzemat en la memòria interna del dispositiu.

+ (NSInteger)getRadius

Obté el Radi d'acció que l'usuari té a la seva configuració, guardada en el terminal.

+ (NSInteger)getMapView

Obté quina és la visualització per als mapes que l'usuari té seleccionada.

+ (NSInteger)getNumDestacados

Obté el número d'ofertes destacades que l'usuari vol que se li mostrin a la pestanya corresponent.

+ (BOOL) setPreferences:(NSInteger)offerReception radius:(NSInteger)radius mapView:(NSInteger)view numberDestacados:(NSInteger)nDestacados

Emmagatzema les preferències especificades com a paràmetres en la memòria interna de l'Iphone. Retornarà un booleà que indicarà si el procés ha anat bé o no.

Marker

Descripció General

Aquesta classe representa els marcadors que s'afegiran al mapa. S'ha hagut d'implementar, per a poder tenir una configuració personalitzada dels marcadors que hi haurà al mapa.

Atributs

coordinate: Coordenades on es col·locarà el Marker dins del mapa.
title: Títol que es mostrarà quan es faci click damunt del Marker.
subtitle: Subtítol que es mostrarà quan es faci click damunt del Marker.

Mètodes

- (id) initWithCoords:(CLLocationCoordinate2D) coords

Mètode que s'utilitza per a inicialitzar un Marker amb unes coordenades determinades.

getters i setters per a title i subtitle

OfferViewController

Descripció General

Aquesta classe és l'encarregada de gestionar l'aspecte visual de la vista d'una oferta. És a dir, és el controlador associat a la vista "OfferView", que és la vista on podem veure la informació concreta d'una oferta.

Classes que utilitza

Oferta: S'utilitzarà una instància de la classe oferta (que en aquest cas, veurem que serà un atribut de la classe) per a saber quina oferta és la que s'ha de mostrar per pantalla.

Atributs

offer: L'oferta que hem de mostrar per pantalla.
llegarButton: Botó de "Cómo Llegar".
backButton: Botó per a tornar a la vista anterior.
offerName: Lloc on es mostrarà el nom de l'oferta.
distance : Lloc on es veurà la distància entre l'usuari i l'oferta.
img : Imatge representativa de l'oferta.
description : Lloc on veurem la descripció

Mètodes

- (IBAction)goBack:(UIButton *)sender

Mètode que es dispara en clicar damunt el backButton i tanca la vista, per a donar pas a la vista que hi havia en pantalla anteriorment.

Vista Associada

Aquí podem veure l'aspecte final de la vista d'una oferta. Podem observar com hi ha dos botons, un per tornar enrere, i un per a veure la ruta que hem de seguir per arribar a l'establiment o oferta de què parlem.

Hi ha una descripció, el nom de l'establiment o oferta i la distància a la qual ens trobem d'aquest establiment. A més, la imatge que es mostra correspon a la font d'on s'ha obtingut.

Com a comentari, cal esmentar que s'han eliminat els botons inferiors dels quals parlàvem en l'apartat del disseny de l'aplicació. Això s'ha portat a terme pel fet que no tenia massa sentit tenir-los si cap d'ells estava seleccionat. S'ha considerat millor fer això amb aquesta vista i la de les rutes, que apareixeran com a elements modals per damunt de les altres vistes.

CustomCellController

Descripció General

Classe que serveix per a personalitzar les cel·les de les llistes del sistema. Per defecte, només se'ns permet mostrar un text a cadascuna. Amb aquesta classe, ho personalitzarem de tal manera, que podrem mostrar diversos textos amb diferents formats i una imatge dins cadascuna de les cel·les de les llistes que mostrarem.

Atributs

titleCell: El títol de la cel·la.
distance : La distància que es mostrarà a la cel·la.
img : La imatge que es mostrarà.

Mètodes

Getters i setters dels atributs

Vista Associada

Aquest és l'aspecte personalitzat que s'ha dissenyat per a cada fila dels nostres llistats. Com a característica diferenciadora respecte a l'aspecte per defecte, podem veure com hi ha una imatge, dos formats diferents de text, i un indicador del fet que aquell element és clicable. Característiques que no ofereix la configuració per defecte. Cal destacar que finalment no s'ha inclòs una petita descripció en la cel·la, ja que recarregava massa la interfície.

MyCLController

Descripció General

Classe que s'encarregarà de gestionar la geolocalització. El seu propòsit és obtenir les coordenades de la posició on és l'usuari en el moment actual.

Atributs

locationManager: Objecte encarregat d'anar obtenint les coordenades de l'usuari.
delegate: Indicador de quina funció es vol executar una vegada es tingui la ubicació de l'usuari.

Mètodes

- (void)locationManager:(CLLocationManager *)manager didUpdateToLocation:(CLLocation *)newLocation fromLocation:(CLLocation *)oldLocation;

Aquesta funció s'executa automàticament quan hi ha un canvi en la posició de l'usuari.

- (void)locationManager:(CLLocationManager *)manager didFailWithError:(NSError *)error;

Funció que s'executa quan en l'obtenció de la posició de l'usuari es produeix un error.

ConfigViewController

Descripció General

Aquesta és la classe que s'encarregarà de gestionar la pestanya "Config", i per extensió, la vista "ConfigView". Quan l'usuari canviï de pestanya, sortint de "ConfigView", s'utilitzarà la classe Preferences per a guardar la nova configuració. Quan l'usuari cliqui sobre "Config", s'haurà de carregar la seva configuració emmagatzemada (o si no en té encara, la configuració per defecte).

Classes que utilitza

Preferences: S'utilitza una classe d'aquest tipus per a la gestió de les preferències de l'usuari. S'agafaran els valors que s'han establert a la vista, i s'emmagatzemaran a la memòria del terminal mitjançant aquesta instància.

Atributs

Radi: Camp de text on es mostra quin radi és el que tenim seleccionat mitjançant l'slider**.**
Destacados : Camp on es mostra quin valor té l'slider dels destacats.
RecepcionOfertas : Botó que activa o desactiva la recepció d'ofertes.
VisualizacionMapas: Botó amb tres parts que permet seleccionar quina visualització es vol per als mapes.
SliderRadi : Slider per a canviar el Radi d'acció.
SliderDestacados : Slider per a canviar el número de destacats.

Mètodes

- (IBAction)radiChanged:(UISlider *)sender;

S'executa quan es canvia el valor de l'Slider. El que fa és posar al camp de text "Radi", el valor actual de l'slider.

- (IBAction)numDestacadosChanged:(UISlider *)sender;

S'executa en el moment en què canviem el valor de l'Slider dels "Destacados", fa la mateixa funció que radiChanged, col·loca el valor de l'slider de destacados, al camp "Destacados" de la interfície.

Vista Associada

Aquest és l'aspecte de la quarta pestanya de l'aplicació. Podem veure els diferents paràmetres que podem editar, com són el radi d'acció, la visualització del mapa o la recepció de les ofertes. Quan entrem a aquesta vista, es carrega la configuració de l'usuari. Una vegada se surt de la vista, guardarem la nova configuració. És un element clau de l'estructura de l'App, ja que repercuteix en totes les altres vistes directament.

DestacatsViewController

Descripció General

Aquesta és la classe que s'encarregarà de gestionar la pestanya "Destacados", i per extensió, la vista "DestacatsView". El que farem serà carregar el número d'ofertes destacades que l'usuari tingui seleccionat a les seves preferències i demanar a l'API aquesta quantitat de resultats. L'API ens retornarà aquesta informació i la mostrarem en un llistat en aquesta vista. Si cliquem a qualsevol de les altres pestanyes, sortirem d'aquesta vista i, per tant, ja no serà aquest el controlador que actuarà.

Classes que utilitza

Preferences: S'utilitza una instància d'aquesta classe per a saber quants elements haurà de tenir la llista de destacats que es mostrarà. El que farem serà carregar de la memòria interna del dispositiu les preferències de l'usuari, i ajustar el número de resultats demanats a l'API en conseqüència.

API: S'utilitzarà la classe de l'API (del costat del servidor), per a obtenir les dades de la base de dades i utilitzar-les en aquesta vista.

Oferta: S'utilitzarà una instància d'aquesta classe per a cadascun dels resultats que l'API retorni. Aquesta classe, la utilitzarem per a tenir els resultats que se'ns han retornat en JSON, en un format molt més fàcil de manipular.

JSON: Per a parsejar els resultats que ens retorna l'API i així poder encapsular-los utilitzant la classe Oferta, haurem de recórrer l'String JSON que se'ns ha retornat. Amb aquesta classe, podem transformar l'String, en el format JSON, i així manipular-lo d'una manera senzilla.

CustomCellController: D'aquesta classe, n'utilitzarem una instància per a cada fila del nostre llistat. Com s'ha explicat abans, aquesta classe té com a objectiu tenir un disseny personalitzat de les files dels nostres llistats.

Atributs

listData: Aquest és l'Array dels objectes de classe Oferta, que s'ha construït a partir del que ens ha retornat l'API en format JSON. És a dir, aquest és l'Array resultant del parseig dels resultats de l'API, que es recorrerà per a renderitzar les dades en format de llista a la vista.

receivedData: Dades que retorna l'API sense parsejar, aquesta serà l'estructura de dades que es recorrerà i s'aniran agafant els seus camps per a generar l'Array d'objectes de què ja hem parlat.

offerList: El llistat que es mostrarà a la interfície. Aquí serà on es mostraran els resultats a nivell visual.

tblCell: Estructura que defineix l'aspecte visual personalitzat de cada fila de la taula offerList. De no haver definit una estructura així, a cada fila, només podríem mostrar text. Així podem mostrar més informació, i en general, un llistat amb més informació.

Mètodes

- (void)get: (NSString *)urlString;

Aquest mètode serà l'encarregat de gestionar la crida que fem a l'API des d'aquesta vista. Farà la crida de manera asíncrona, a la URL que li passem com a paràmetre, i obtindrà les dades del servidor. Recordem que el número de destacats que demanarem serà el que l'usuari tingui establert a les seves preferències i estarà com a paràmetre de la URL.

- (void)ofertesDestacades:(NSString *)data;

Aquest mètode rebrà la informació que ens ha arribat del servidor, i, mitjançant la classe externa JSON, recorrerà les dades i formarà l'Array d'objectes Oferta del qual hem parlat abans.

- (void)renderData:(NSMutableArray *)data;

En aquest punt, rebem l'Array d'ofertes ja format i amb aquest mètode el que fem és agafar aquestes dades i mostrar-les al llistat offerList.

Vista Associada

Aquest és l'aspecte de la llista dels destacats del sistema. Se'n mostren tants com l'usuari ha seleccionat a la configuració, número que surt damunt de la pestanya.

Com veiem, si la recepció d'ofertes està desactivada (cosa que podem fer des de la vista de configuració), quan entrem en aquesta vista, se'ns informa d'aquest fet.

ProperesViewController

Descripció General

Aquesta és la classe que s'encarregarà de gestionar la pestanya "Cerca de Ti", la segona de l'aplicació. La vista que gestiona aquest controlador s'anomena "ProperesView". El funcionament d'aquesta part de l'App és molt senzill: quan es clica a la pestanya esmentada, s'agafa la posició de l'usuari, i es fa una crida a l'API per a obtenir els resultats que són dins del radi d'acció definit per l'usuari a les seves preferències. Els resultats es mostraran en format llista, ordenades per proximitat a la posició de l'usuari. Si es clica damunt una de les files, s'obrirà la vista "OfferView", gestionada per l' "OfferViewController" de què hem parlat abans.

Classes que utilitza

Preferences: S'utilitza una instància d'aquesta classe per a saber amb quin radi d'acció, volem demanar els resultats a l'API. Recordem que li passarem la posició de l'usuari i el radi, per a veure quins establiments o ofertes hi ha dins d'aquest radi.

API: S'utilitzarà la classe de l'API (del costat del servidor), per a obtenir les dades de la base de dades i utilitzar-les en aquesta vista.

Oferta: S'utilitzarà una instància d'aquesta classe per cada un dels resultats que l'API retorni. Aquesta classe la utilitzarem per a tenir els resultats que se'ns han retornat en JSON, en un format molt més fàcil de manipular.

JSON: Per a parsejar els resultats que ens retorna l'API, i així poder encapsular-los utilitzant la classe Oferta, haurem de recórrer l' String JSON que se'ns ha retornat. Amb aquesta classe, podem transformar el String, en el format JSON, i així manipular-lo d'una manera senzilla.

CustomCellController: D'aquesta classe, n'utilitzarem una instància per a cada fila del nostre llistat. Com s'ha explicat abans, aquesta classe té com a objectiu tenir un disseny personalitzat de les files dels nostres llistats.

MyCLController: Aquest controlador s'utilitzarà per a obtenir la posició de l'usuari.

Atributs

listData: Aquest és l'Array dels objectes de classe Oferta, que s'ha construït a partir del que ens ha retornat l'API en format JSON. És a dir, aquest és l'Array resultant del parseig dels resultats de l'API, que es recorrerà per a renderitzar les dades en format de llista a la vista.

receivedData: Dades que retorna l'API sense parsejar, aquesta serà l'estructura de dades que es recorrerà i s'aniran agafant els seus camps per a generar l'Array d'objectes del qual ja hem parlat.

offerList: El llistat que es mostrarà a la interfície. Aquí serà on es mostraran els resultats a nivell visual.

tblCell: Estructura que defineix l'aspecte visual personalitzat de cada fila de la taula offerList. Si no s'hagués definit una estructura així, a cada fila només podríem mostrar text. Així podem mostrar més informació i, en general, un llistat amb més informació.

locationController: Controlador que serà l'encarregat d'obtenir la posició de l'usuari.

latitude: Latitud geogràfica de la posició de l'usuari. Obtingut amb el locationController.

longitude: Longitud geogràfica de la posició de l'usuari. Obtingut amb el locationController.

Mètodes

- (void)get: (NSString *)urlString;

Aquest mètode serà l'encarregat de gestionar la crida que fem a l'API des d'aquesta vista. Farà la crida de manera asíncrona, a la URL que li passem com a paràmetre, i obtindrà les dades del servidor. Recordem que a l'API enviarem (com a paràmetres de la URL), la posició de l'usuari i el seu radi d'acció.

- (void)ofertesProperes:(NSString *)data;

Aquest mètode rebrà la informació que ens ha arribat del servidor, i, mitjançant la classe externa JSON, recorrerà les dades i formarà l'Array d'objectes Oferta del qual hem parlat abans.

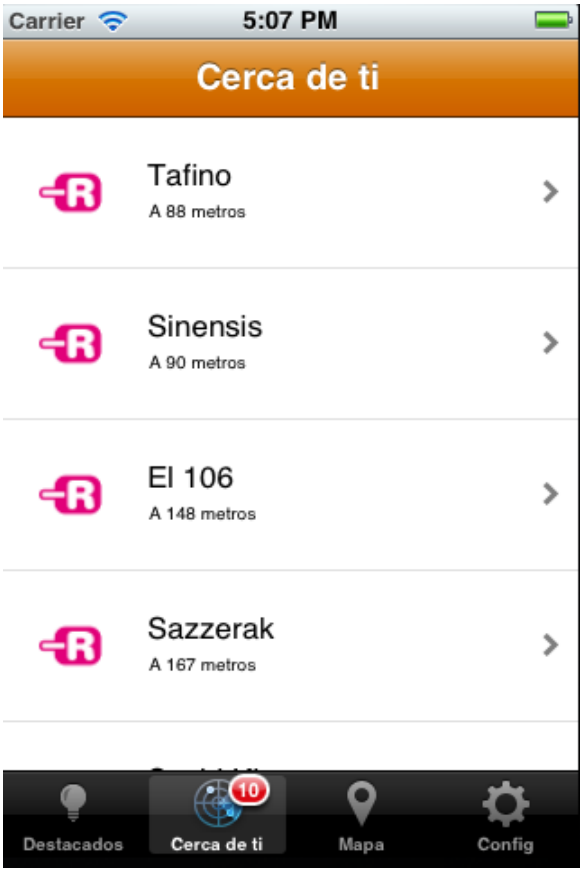
- (void)renderData:(NSMutableArray *)data;

En aquest punt, rebem l'Array d'ofertes ja format i, amb aquest mètode, el que fem és agafar aquestes dades i mostrar-les al llistat offerList.

- (void)locationUpdate:(CLLocation *)location;

Mètode que setejarà els atributs latitude i longitude. És a dir, serà el mètode encarregat d'obtenir la posició de l'usuari, utilitzant el controlador que tenim com a atribut de la classe.

Vista Associada



Aquí podem veure l'aspecte de la vista de les ofertes que hi ha al teu voltant. Podem veure que, si tenim la recepció d'ofertes desactivada, se'ns informa.

Clicant damunt de qualsevol dels elements del llistat, ens portarà a la vista d'aquell element en concret. Veiem que hi ha el número d'ofertes que pots trobar dins el teu radi d'acció (el que podem canviar des de la configuració) damunt de la pestanya.

Observem també que les ofertes se'ns mostren per ordre ascendent de distància, per tal de prioritzar aquelles que són més a prop de l'usuari.

MapViewController

Descripció General

Aquest és el controlador que gestiona la pestanya “Mapa” de l’aplicació. És l’encarregat de gestionar la vista “MapView”. La seva funció és mostrar un mapa on aparegui la posició de l’usuari i, com a marcadors, les ofertes i la informació de restaurants, a la seva posició. D’aquesta manera, l’usuari veurà on estan localitzats tant els restaurants com les ofertes que tenim a la base de dades.

Classes que utilitza

Preferences: S’utilitza una instància d’aquesta classe per a saber quin és el tipus de mapa que l’usuari vol veure. Pot triar entre tres visualitzacions diferents: "Mapa", "Satelite" i "Hibrido", cada una amb el seu propi aspecte. Recordem que la configuració d’aquest aspecte es farà amb la pestanya “Config” de la nostra App.

API: S’utilitzarà la classe de l’API (del costat del servidor), per a obtenir les dades de la base de dades i utilitzar-les en aquesta vista. En concret, el que farem serà obtenir tota la informació que tenim a la base de dades per a representar-la en forma de marcadors.

Oferta: S’utilitzarà una instància d’aquesta classe per a cadascun dels resultats que l’API retorni. Aquesta classe la utilitzarem per a tenir els resultats que se’ns han retornat en JSON, en un format molt més fàcil de manipular.

JSON: Per a parsejar els resultats que ens retorna l’API, i així poder encapsular-los utilitzant la classe Oferta, haurem de recórrer l’String JSON que se’ns ha retornat. Amb aquesta classe, podem transformar l’String en el format JSON, i així manipular-lo d’una manera senzilla.

Marker: Aquesta classe s’utilitzarà per a gestionar els marcadors. Tindrem una instància per a cadascun dels marcadors que col·loquem al mapa.

Atributs

listData: Aquest és l’Array dels objectes de classe Oferta, que s’ha construït a partir del que ens ha retornat l’API en format JSON. És a dir, aquest és l’Array resultant del parseig dels resultats de l’API, que recorrerem per a inicialitzar tots els Markers.

receivedData: Dades que retorna l’API sense parsejar, aquesta serà l’estructura de dades que es recorrerà i s’aniran agafant els seus camps per a generar l’Array d’objectes del qual ja hem parlat.

map: Element de la interfície que representa el mapa. A sobre s'hi col·locaran els Markers.

markerInfo: Array de marcadors que es formarà a partir del vector "listData" i que recorrerem a l'hora de col·locar els punts al mapa.

Mètodes

- (void)get: (NSString *)urlString;

Aquest mètode serà l'encarregat de gestionar la crida que fem a l'API des d'aquesta vista. Farà la crida de manera asíncrona, a la URL que li passem com a paràmetre, i obtindrà les dades del servidor.

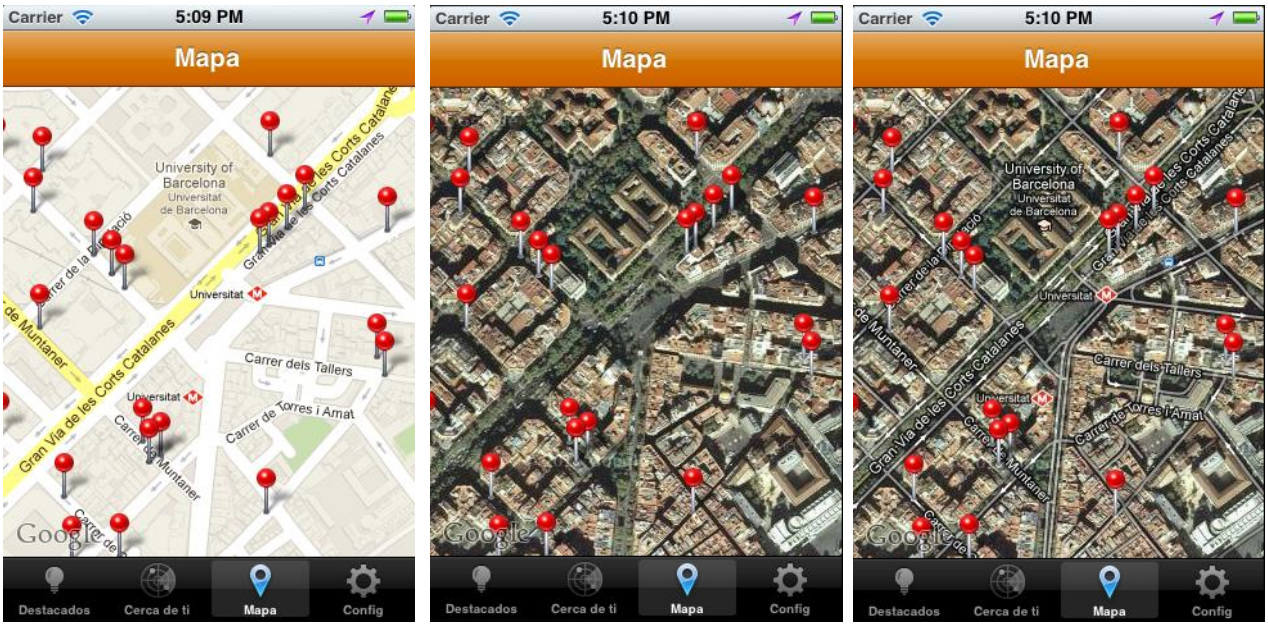
- (void)buildMarkerArray:(NSString *)data;

Aquest mètode rebrà la informació que ens ha arribat del servidor i, mitjançant la classe externa JSON, recorrerà les dades i formarà l'Array de Markers, on tindrem la informació dels marcadors que col·locarem posteriorment.

- (void)addMarkers:(NSMutableArray *)data;

En aquest punt, rebem l'Array de Markers ja format i, amb aquest mètode, el que fem és agafar aquestes dades i mostrar-les al mapa.

Vista Associada



Aquí veiem l'aspecte de la vista del mapa, on es mostren tots els establiments i ofertes a la seva posició geogràfica. Podem observar les tres visualitzacions possibles que s'ofereixen.

JSON

Descripció General

Classe externa utilitzada per a gestionar els Strings JSON que ens retorna el servidor. No entraré més en detall, ja que no és una classe que hagi implementat personalment.

RouteViewController

Descripció General

Aquest és el controlador de la vista que al nostre sistema s'anomena "RouteView", la qual mostra a l'usuari la ruta que hauria de seguir per a arribar des de la seva posició actual, a la posició de l'oferta o establiment que ha seleccionat anteriorment. Recordem que per arribar a aquesta vista, hem d'haver arribat a la vista d'una oferta (ja sigui des de la vista de "Destacados", de la de "Cerca de ti" o bé clicant damunt d'un dels marcadors del mapa), i haver clicat al botó que té com a text "Cómo Llegar". Una vegada fet això, es mostrarà un mapa amb la posició de l'usuari, la de l'oferta seleccionada i una ruta entre els dos punts. Per a obtenir aquesta ruta, s'ha cridat a l'API de Google Directions[42] i s'ha representat mitjançant rectes damunt del mapa la ruta que ens ha retornat.

Classes que utilitza

Marker: Com a la vista del mapa, de la qual hem parlat abans, necessitem utilitzar objectes d'aquesta classe, per a representar els marcadors personalitzats que hem creat. En aquest cas, n'hi haurà dos, un per a l'usuari i un per a l'oferta o establiment.

JSON: Com sempre que fem crides a una API (ja sigui la nostra pròpia o una d'externa), si els resultats són en format JSON, haurem d'utilitzar aquesta classe per a manipular les dades.

Preferences: La utilitzarem perquè el mapa es mostri amb l'aspecte que l'usuari ha seleccionat a les seves preferències.

MyCLController: Com en altres casos, aquesta classe serà la que ens permetrà obtenir la posició de l'usuari.

Atributs

map: Element de la interfície que representa el mapa. A sobre es col·locaran els Markers i també es dibuixarà la ruta.

[42] **Google Directions:** Servei que ofereix google, que donats dos parells de coordenades geogràfiques et calcula una ruta entre els dos punts.

latitudeUser: Component que correspon a la latitud de l'usuari.

longitudeUser: Component que correspon a la longitud de l'usuari.

latitudeMarker: Latitud de l'oferta que, en aquesta vista, serà el destí de la ruta que surt de la posició de l'usuari.

longitudeMarker: Longitud de l'oferta.

receivedData: String sense parsejar, que ens retornarà l'API de Google Directions.

Mètodes

- (IBAction)goBack:(id)sender;

Acció que saltarà quan l'usuari cliqui damunt el botó de "back" i que retornarà a la vista anterior.

- (void)get: (NSString *)urlString;

Aquest mètode serà l'encarregat de gestionar la crida que fem a l'API de Google Directions des d'aquesta vista. Farà la crida de manera asíncrona, a la URL que li passem com a paràmetre, i obtindrà les dades referents a com arribar a la posició de l'oferta, partint des de la posició de l'usuari.

- (void)parseRoute:(NSString *)jsonUnparsed;

Aquesta funció s'executarà quan s'hagi obtingut el resultat de la crida a Google DIrections. El que farà serà, a partir del retornat, extreure el que s'anomena la "overview_polyline", que en poques paraules, és una manera que té google de codificar una ruta entre dos punts. Una vegada obtingut aquest camp, es cridarà a polylineWithEncodedString, que farà la descodificació pròpiament dita.

- (void)renderRoute;

Una vegada obtingut l'objecte MKPolyline, l'haurem de mostrar damunt del mapa. Això és el que farà aquest mètode.

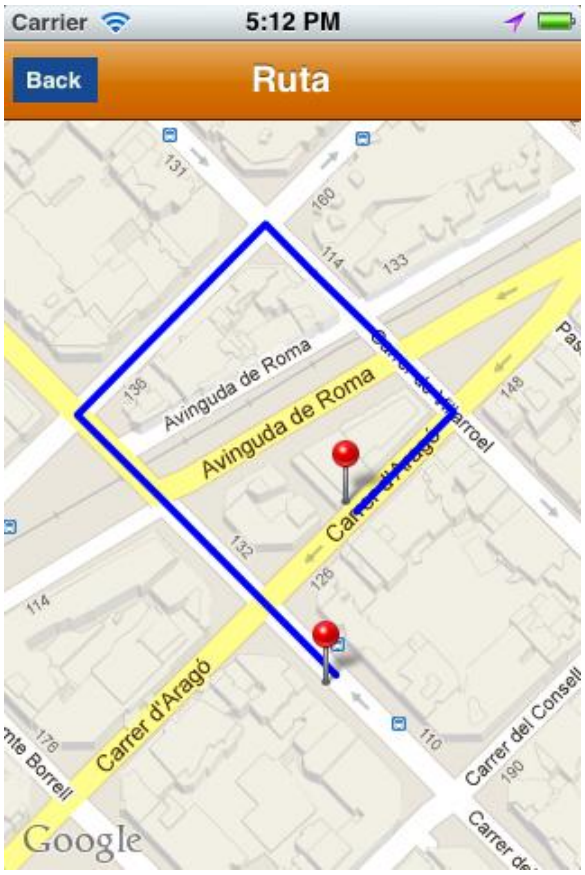
-(void) getRoute;

Aquest mètode construirà la URL a la qual cridarà el mètode "get". En aquesta URL se li haurà de passar, entre d'altres coses, el punt d'inici de la ruta (les coordenades) i el de fi. Una vegada construïda la URL, es cridarà a la funció get, de la qual ja hem parlat.

+ (MKPolyline *)polylineWithEncodedString:(NSString *)encodedString;

En aquest punt, ja hem extret l'"overview_polyline" retornada per Google. Aquí la transformarem en un objecte MKPolyline, objecte que ja podrem manipular per a mostrar-lo a l'usuari.

Vista Associada



Aquesta vista és, sense cap dubte, la que més dificultats ha suposat quant a implementació. Tot i així, podem veure com s'ha aconseguit l'objectiu: mostrar les rutes que cal seguir per a arribar a un punt en concret del mapa, des de la teva posició. No s'han mostrat els seus tres aspectes possibles, però podem veure com el canvi de visualització dels mapes a la configuració també afecta a aquesta vista.

Implementació del projecte: API

Com hem comentat anteriorment, l'aplicació es comunicarà amb el servidor mitjançant una classe que hi ha al mateix servidor. Aquesta classe es va especificar a l'apartat de especificació, i ara es mostrarà com accedir-hi i, en general, com s'ha implementat. Ja s'ha comentat que la part del servidor es programaria en Php i que aquesta classe seria qui retornaria les dades a l'aplicació, utilitzant el format JSON.

Un exemple de retorn de l'API el podem veure a continuació:


```
{
   "id":"1431",
   "id_restalo":"522",
   "lat":"41.387325",
   "lng":"2.113446",
   "description":"El establecimiento Julivert Meu es un oasis dentro
         del bullicio de una gran ciudad. Situado en el distrito
         de Les Corts, este restaurante es un lugar rústico sin lujos
         ni pretensiones, con mesas de madera y jarras de barro.
   "name":"Julivert Meu",
   "address":"Jordi Girona, 12",
   "link":"http://www.restalo.es/restaurante-julivert-meu",
   "source":"Restalo",
    "score":"851",
   "distance":12.125452144522
}
```


En aquest cas, només ens ha retornat una oferta. Aquest String serà interpretat i parsejat per l'aplicació de manera senzilla (utilitzant la classe JSON de la qual hem parlat abans).

Per tal que el servidor ens retorni aquesta informació, s'haurà de fer una crida a una URL en concret. El que es fa realment és accedir a un mòdul del servidor, que parseja la URL, i extreu els paràmetres. Si son correctes, els utilitza per a cridar les funcions que s'indiquin de la classe.

A continuació, es mostren les URL a les quals es pot fer crides, per a obtenir informació de la nostra base de dades. En totes les crides, hi ha un paràmetre "q", que serà quina de les funcions es vol executar. Els altres paràmetres seran els que es passaran a la funció especificada a la "q".

Obtenir les ofertes del teu voltant:

`http://qa.apps3.com/API/api.php?q=obtenirProperes&lat=41.387325&lng=2.113446&rad=100`

Paràmetres que passem al servidor:

- q : Funció que es vol executar
- lat: Latitud de la posició l'usuari
- lng: Longitud de la posició de l'usuari
- rad: Radi d'acció de l'usuari

Com Funciona internament:

El funcionament d'aquesta funció és molt senzill, fa comparacions entre la distància que hi ha entre la posició passada per paràmetre i les posicions dels elements de la base de dades. Si aquesta distància és menor o igual al radi d'acció, s'agafa el resultat i s'afegeix a un Array, que posteriorment serà passat a format JSON. Una vegada obtingudes totes les ofertes, s'ordenen per ordre ascendent respecte a la distància, per tal de retornar els resultats de més a prop a més enfora de l'usuari. Per a ordenar-los, s'ha utilitzat l'algoritme del Merge Sort [43].

Per a calcular les distàncies en metres, entre dues coordenades geogràfiques, utilitzo la funció següent:

```
private function distancia($lat1, $long1, $lat2, $long2){

      $degtorad = 0.01745329;
      $radtodeg = 57.29577951;

      $dlong = ($long1 - $long2);
      $dvalue = (sin($lat1 * $degtorad) * sin($lat2 * $degtorad))
        + (cos($lat1 * $degtorad) * cos($lat2 * $degtorad)
        * cos($dlong * $degtorad));

       $dd = acos($dvalue) * $radtodeg;

       $miles = ($dd * 69.16);
       $km = ($dd * 111.302);

       return $km * 1000;
}
```

[43] **Merge Sort**: Algoritme d'ordenació molt conegut basant en el principi de "divide y vencerás", té una complexitat de O(n log n) i és considerat un dels algoritmes d'ordenació més eficients. Com a curiositat, l'autor d'aquest algoritme va ser John Von Neumann.

Obtenir les dades d'una oferta

```
http://qa.apps3.com/API/api.php?q=obtenirOferta&kind=Restalo&name=El%20Pati%20Blau
```

Paràmetres que passem al servidor:

- q : Funció que es vol executar
- kind: A quina font d'ofertes pertany (Restalo, Groupon, Groupalia o El Tenedor)
- name: Nom de l'oferta.

Com funciona internament:

Aquesta funció simplement busca la informació d'una oferta en concret. Com que totes les ofertes tenen noms diferents, simplement es fa una query que retornarà un resultat, i en JSON.

Obtenir les ofertes destacades

```
http://qa.apps3.com/API/api.php?q=obtenirDestacats&n=5
```

Paràmetres que passem al servidor:

- q : Funció que es vol executar
- n: Número d'ofertes destacades que volem que se'ns retornin.

Com funciona internament:

Aquesta funció agafarà els resultats de la base de dades que tinguin el camp "score" amb un valor més alt. S'agafaran els "n" primers, essent "n" el paràmetre que es passarà per URL i que serà el valor que l'usuari té a les seves preferències.

El camp score s'ha calculat agafant la puntuació interna que tenien totes les fonts d'ofertes de les quals hem parlat, i normalitzant els valors, per tal de tenir les dades en una escala comuna.

Implementació del projecte: Base de Dades

Tot i ser la part menys esmentada, la base de dades és crucial per al funcionament de l'aplicació, ja que és el que proporciona contingut i dóna sentit realment a tot el projecte. Si no hi hagués aquesta base de dades l'aplicació no tindria cap tipus d'utilitat.

Com hem comentat en apartats anteriors, la tecnologia que s'ha utilitzat per a la base de dades és MySql, que tot i no ser tan potent com altres alternatives, com PostgreSQL, era la més senzilla de configurar, gràcies a la meva experiència prèvia amb la tecnologia.

Les ofertes i informació sobre establiments que consulta l'App són a una taula anomenada "Offers" i que té una estructura com aquesta:

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
☐	id	int(11)			No	*None*	AUTO_INCREMENT
☐	id_restalo	int(11)			Sí	*NULL*	
☐	lat	double			No	*None*	
☐	lng	double			No	*None*	
☐	description	varchar(255)	utf8_general_ci		No	*None*	
☐	name	varchar(255)	utf8_general_ci		No	*None*	
☐	address	varchar(255)	utf8_general_ci		Sí	*NULL*	
☐	link	varchar(255)	utf8_general_ci		No	*None*	
☐	source	varchar(255)	utf8_general_ci		No	*None*	
☐	score	int(11)			No	*None*	

Com podem veure, no hi ha massa per comentar, tenim un identificador únic per fila i la resta de columnes són les previsibles, i de les quals ja hem parlat anteriorment, com són la latitud, longitud, títol, o descripció de les ofertes o establiments. En el nostre cas, aquesta configuració ha estat més que suficient per al desenvolupament del projecte, ja que només teníem unes 4000 files en aquesta taula. Com que això acabarà creixent en el projecte de l'empresa, el que es té pensat fer és fer cerques mitjançant l'Sphinx[44], que permet fer cerques específiques per coordenades i en general milloraria molt el rendiment d'aquesta part del sistema.

[44] **Sphinx:** Motor de cerca dissenyat per a indexar continguts de bases de dades. Actualment es pot utilitzar sobre MySQL, PosgreSQL i altres bases de dades. Permet aconseguir una reducció del temps de cerca important, respecte al que s'obté aplicant un indexat tradicional.

Problemes que han sorgit

Una vegada acabat el procés d'implementació, es pot parlar dels problemes que han sorgit en aquesta fase.

El primer problema era previsible. Inicialment es partia de zero en la programació en Objective C per a dispositius amb iOS. Al principi va ser complicat acostumar-se, però amb paciència, i moltes hores fent proves, s'ha aconseguit aprendre el llenguatge. Com a comentari, s'ha de destacar que no només s'ha après, sinó que s'ha fet amb molt de gust, ja que tot i tenir peculiaritats, l'experiència ha estat molt positiva.

Per altra banda, ha sorgit el problema amb les rutes cap a les ofertes. Donat que els dispositius amb iOS utilitzen els mapes de Google Maps, vaig assumir que els càlculs de rutes estarien implementats dins l'SDK, tal i com estan en Android. En aquest punt vaig veure que la gent de Google, no ho deixen tan fàcil als programadors d'Apple. Per a solucionar això, s'ha fet mitjançant l'API de Google Directions, fent crides i aconseguint després d'una mica d'investigació descodificar de manera eficient el retorn.

En el procés de recopilació d'ofertes, ja fos amb el crawler o amb l'API de Restalo, van sorgir problemes amb la codificació dels caràcters, pel fet que, en moltes ocasions, el filtratge aplicat destruïa les dades, a causa de la seva mala codificació en les webs consultades. Ajustant millor el filtratge, s'han aconseguit resultats acceptables.

A part d'aquests detalls, no hi ha hagut grans problemes, gràcies a la documentació d'Apple, que és molt completa, i té tutorials que expliquen de manera pràctica totes les funcionalitats que ofereix el seu SDK.

Resultats

Una vegada acabat el procés de desenvolupament de l'aplicació, es poden analitzar els resultats obtinguts. S'ha implementat una aplicació totalment funcional, que s'aprofita del contingut de la base de dades i el representa d'una manera amigable a l'aplicació mòbil, que segueix l'estètica típica de les App per a dispositius iOS.

L'aplicació es comporta d'una manera molt fluïda, inclús en els casos en què demana moltes dades al servidor. En aquests casos, s'aprecia un petit retard, però totalment acceptable, pel volum de dades.
El càlcul de rutes funciona d'una manera precisa i, en general, l'aplicació es comporta tal i com esperàvem a l'inici del procés de desenvolupament.

S'ha complert amb les fites marcades prèviament i, com a conclusió, podem dir que l'aplicació compleix els requeriments demanats a l'inici. S'han descartat les funcionalitats opcionals de les quals parlàvem per falta de temps.
S'ha provat tant amb el simulador que ofereix l'entorn de desenvolupament, com amb dispositius reals.

Si comparem el rendiment que s'ha obtingut en aquest cas amb l'experimentat en ocasions anteriors de la meva vida professional, amb algunes de les tecnologies estudiades al primer apartat, es pot observar com el codi natiu es comporta d'una manera molt millor i dóna un rendiment molt superior. Per tant es podria concloure que s'ha fet una bona elecció.

Comentaris sobre la planificació

Una vegada acabat el projecte, podem parlar de l'execució de la planificació que s'ha mostrat a l'inici del document.

S'ha seguit la planificació, però aquesta s'ha vist una mica alterada per alguns factors que s'han sobreestimat o subestimat.

Per posar algun exemple, la fase de disseny i especificació s'ha allargat més del previst, a causa de la gran quantitat de documentació que s'ha generat. En canvi, la fase d'aprenentatge i d'implementació de la part del servidor s'ha fet en menys temps del previst i, en general, la fase d'implementació s'ha anat fent ràpidament, menys en el cas del càlcul de rutes, que s'ha presentat com una funcionalitat menys trivial del que s'havia previst inicialment.

En conclusió, tot i no seguir-se de manera estricta, hi ha hagut tasques que s'han escurçat i algunes que s'han allargat, però al final s'ha acabat equilibrant i no hi ha hagut una gran desviació quant a hores totals.

Conclusions

En el desenvolupament d'aquest projecte, moltes són les coses que s'han après. Vist en retrospectiva, podem extreure una sèrie de punts clau, que considero que són les conclusions amb les quals ens hem de quedar, una vegada acabat el projecte:

Dels estudis previs, podem destacar que hem descobert que els usuaris d'Apple són més curiosos quan se'ls esmenta pel Twitter i se'ls ofereixen ofertes properes a la seva posició.

Hem vist també que la utilització d'Smartphones per a twittejar de manera geolocalitzada és més freqüent en les zones cèntriques de la ciutat de Barcelona.
Podríem dir que el Twitter és una bona font per a extreure informació del que està passant al teu voltant, gràcies al fet que molts Tweets estan vinculats a una localització particular. És una font que es pot explotar molt, i que fent una mica de mineria de dades, es poden fer uns estudis sobre el comportament de la gent molt interessants.

Una de les coses que més m'han impactat dels estudis previs, ha estat que el trànsit de bots que hi ha pel Twitter és quasi comparable al d'usuaris reals. Relacionat amb això, també s'ha vist com Twitter té un sistema bo per a la detecció de missatges emesos de manera automàtica, però que no és ni molt menys infalible amb una mica d'investigació i prova.

Quant a la recopilació d'ofertes geolocalitzades, s'ha demostrat que recopilar-les mitjançant un crawler és factible, tot i que comporta alguns problemes (sobretot de codificació de caràcters).

Personalment, he vist que programar per a dispositius iOS és una experiència molt enriquidora i amb una corba d'aprenentatge bastant assequible, a causa de la gran quantitat de documentació disponible i d'una comunitat de desenvolupadors molt activa.

Relacionant aquest fet amb el projecte propi de l'empresa, aquest PFC ha estat una eina molt útil per a fer una mica de recerca, tant referent a tecnologies com per descobrir quin segment del mercat era el més apropiat. Es considera el projecte com un prototip, que serà el punt d'inici per al desenvolupament del producte final. També ha estat una manera molt efectiva d'aprendre una tecnologia que hores d'ara cap membre de l'empresa coneixia.

En definitiva, ha estat una experiència molt enriquidora en tots els aspectes, tant professionals, acadèmics com en l'àmbit personal.

Glossari

Web Crawler: Programa que inspecciona les pàgines web de forma metòdica i automatitzada.

Php: Llenguatge de programació interpretat, dissenyat originalment per a crear pàgines web dinàmiques. S'utilitza principalment per a la interpretació del costat del servidor (server-side scripting).

Symfony: És un complet Framework php, dissenyat per a optimitzar el desenvolupament d'aplicacions web, que proporciona diverses eines i classes encaminades a reduir el temps de desenvolupament d'una aplicació web complexa.

Framework: Estructura conceptual i tecnològica de suport definit, normalment mitjançant artefactes o mòduls de software concrets, en base als quals, un altre projecte de software pot ser més fàcilment organitzat i desenvolupat.

API (Application Programming Interface)*:* Conjunt de funcions i procediments que ofereix certa llibreria per a ser utilitzada per un altre software com a capa d'abstracció. Moltes pàgines web ofereixen una API pública, perquè altres desenvolupadors utilitzin la informació que tenen emmagatzemada.

Xarxa Social: Estructures socials compostes per grups de persones, que estan connectades per diferents tipus de relacions, com puguin ser interessos, amistat, parentesc o coneixements comuns. Alguns exemples de xarxes socials serien Facebook, Twitter, Youtube, Myspace, Linkedin, Instagram...

Microblogging: Servei que permet als usuaris enviar i publicar missatges breus, generalment de text pla únicament.

Sistema de Achievements: Sistema que implementen moltes xarxes socials i jocs actuals, on es premia als usuaris segons les accions que fan amb el programa en concret. En el món dels videojocs es sol conèixer com "Trofeus" i en les xarxes socials es pot veure fàcilment amb el sistema de medalles de Foursquare, o en el sistema dels jocs de Facebook, on et recompensen amb punts del mateix joc, si has complert una sèrie d'objectius

Crontab: Fitxer que en entorns Linux s'utilitza per a gestionar processos automatitzats. Afegir un procés automatitzat al sistema és tan fàcil com afegir una línia en aquest fitxer indicant quina comanda s'ha d'executar, en quins dies i a quines hores.

Marker: En el context de l'API de Google Maps, un Marker, o marcador, és un indicador que es col·loca en una posició concreta del mapa, per a representar algun tipus d'informació. En el nostre cas, cada Marker mostra la posició geogràfica d'un Tweet i, si cliques damunt, pots veure el Tweet en qüestió.

Menció: Al Twitter és un Tweet que va dirigit a un usuari o usuaris en concret, que té com a estructura: @[user1]...@[userN] [Missatge]. Aquests missatges es mostraran en els perfils dels usuaris esmentats.

Llibreria Curl: Llibreria en aquest cas de php, que simula el comportament d'un navegador web.

Aplicacions de Twitter: Al Twitter, són les encarregades de connectar amb el perfil d'un usuari, de tal manera que si es volen automatitzar accions, s'haurà d'enregistrar una aplicació vinculada amb el perfil al qual es vulgui actuar.

Direcció IP: Etiqueta numèrica que identifica de manera lògica i jeràrquica una interfície d'un dispositiu dins d'una xarxa IP.

User-Agent: En aquest cas, em refereixo al camp de les capçaleres HTTP que s'envien quan es fa una petició. Al camp l' User-Agent, tenim informació de l'aplicació, la versió, el sistema operatiu i l'idioma de l'usuari que ha fet la petició.

Banejar: Expulsar o eliminar el dret d'accedir a un determinat servei per una violació de les normes d'utilització del servei en concret.

Web Proxy: Es tracta d'un tipus d'aplicacions web que fan d'intermediari en les peticions que fa l'usuari. D'aquesta manera, s'aconsegueix que la IP de l'usuari, per al receptor, passi a ser la del Web Proxy.

Javascript: Llenguatge de programació interpretat, que s'utilitza principalment al costat del client, implementat com a part del navegador web de l'usuari. Permet la creació de pàgines web dinàmiques amb interfícies molt elaborades. S'utilitza també en molts altres contextos.

Mysql: Sistema gestor de bases de dades relacional molt popular actualment. Compta amb controladors específics per a gran quantitat de llenguatges de programació, com php, C, C++, C#, Pascal, Java, Lisp, Perl, Python...

CSS: Llenguatge utilitzat per a definir l'aspecte d'un document escrit en HTML o XML.

WebOs: Sistema operatiu mòbil, inicialment desenvolupat per Palm Inc, ara propietat d'HP. Inicialment era considerat un sistema amb moltes possibilitats, però HP va deixar de desenvolupar dispositius amb aquest sistema i ha alliberat el codi font del sistema.

Symbian: Sistema operatiu per a dispositius mòbils, utilitzat sobretot en dispositius Nokia. Era un dels sistemes més utilitzats abans que iOS i Android s'establissin com les dues grans potències dins de les tecnologies mòbils actuals.

Bada: Sistema operatiu mòbil desenvolupat per Samsung, molt minoritari si el comparem amb iOS, Android, o inclús Symbian.

Plug-in: Es tracta d'una aplicació que es relaciona amb una altra per a aportar una funcionalitat nova i molt específica.

Malware: Terme genèric que engloba tot software dissenyat amb propòsits maliciosos.

IDE (*Integrated Development Environment*): És un programa informàtic compost per un conjunt d'eines de programació. Pot ser tant per un llenguatge en concret, com per diversos. Solen comptar amb un editor de codi, un compilador i eines per a debugar codi generalment. Entre els IDE que podríem destacar hi ha Netbeans, Eclipse, Xcode o el Visual Studio.

Eclipse: IDE de codi obert multiplataforma i multillenguatge. Té un sistema molt potent de plug-ins, que permet instal·lar components específics i personalitzar les funcionalitats del teu IDE depenent de les teves necessitats com a desenvolupador.

Emulador: En aquest cas, es tracta d'un programa que forma part del plug-in d'Android de l'Eclipse, que emula el comportament d'un dispositiu mòbil amb Android. És totalment configurable i s'utilitza per a agilitzar la programació i no haver d'estar provant contínuament el codi amb dispositius reals.

Terminals de gamma mitjana: En aquest cas, ens referim als Smartphones, que, tot i no tenir grans especificacions tècniques, compleixen amb les seves funcionalitats bàsiques, i tenen un cost inferior als terminals de gamma alta. Tenen una quota molt alta dins el mercat.

Instagram: És una aplicació gratuïta per a dispositius Apple que permet compartir les fotos fetes amb el teu Smartphone. Compta amb una sèrie de filtres fotogràfics per a editar les fotografies i es connecta amb les principals xarxes socials actuals. Actualment, té més de 12 milions d'usuaris i es calcula que s'han compartit més de 100 milions de fotos.

Smalltalk: Llenguatge de programació on la interacció dels objectes es fa mitjançant l'enviament de missatges.

Xcode: IDE desenvolupat per Apple, gratuït, i instal·lat de sèrie amb Mac OS, que permet compilar codis de diversos llenguatges com C, C++, Objective-C o Java. Es tracta d'un IDE molt potent, especialment recomanable per a programar aplicacions mòbils.

Drag & Drop: Expressió que en informàtica es refereix a arrossegar elements d'una finestra a una altra o de moure els elements dins una mateixa finestra.

Virtualitzar: En aquest cas, ens referim a emular un altre sistema operatiu, executant-se mitjançant un programa com VIrtualbox, que alhora s'executa en el marc d'un sistema operatiu en concret. És a dir, ens referim a executar un sistema operatiu dins un altre sistema operatiu, mitjançant un software de virtualització.

JSON: Format lleuger d'intercanvi de dades. Va sorgir per a millorar el rendiment dels intercanvis de dades, que fins al moment es feien en XML, format molt potent, però molt pesat.

Query: Consulta a una base de dades, amb la qual s'obtenen dades.

Google Directions: Servei que ofereix Google que amb dos parells de coordenades geogràfiques et calcula una ruta entre els dos punts.

Merge Sort: Algoritme d'ordenació molt conegut basat en el principi de "divide y vencerás". Té una complexitat de O(n log n) i és considerat un dels algoritmes d'ordenació més eficients. Com a curiositat, l'autor d'aquest algoritme va ser John Von Neumann.

Sphinx: Motor de cerca dissenyat per a indexar continguts de Bases de dades. Actualment es pot utilitzar sobre MySQL, PosgreSQL i altres bases de dades. Permet aconseguir una reducció del temps de cerca important, respecte al que s'obté aplicant un indexat tradicional.

Links Recomanats

L'Smart Shopper

http://montsemarketing.wordpress.com/tag/smart-shopper/

http://gigaom.com/2011/10/31/mobile-and-the-rise-of-the-smart-buyer/

PlaceCast

http://placecast.net/

Appstylus SL i Barcelona Activa

http://www.appstylus.com/

http://www.barcelonactiva.cat/barcelonactiva/cat/

Goutte

http://www.phparch.com/2010/04/four-new-php-5-3-components-and-goutte-a-simple-web-scraper/

Estudis previs

http://qa.apps3.com/chollos/web/app_dev.php/map/tweets/1/0

http://qa.apps3.com/chollos/web/app_dev.php/map/tweets/1/23-03

http://qa.apps3.com/chollos/web/app_dev.php/map/tweets/0/23-03

Sobre iOS

http://www.theinquirer.es/2012/05/08/ios-sigue-a-la-cabeza-entre-los-desarrolladores.html

Fragmentació Android

http://developer.android.com/resources/dashboard/platform-versions.html